305-CD-023-002

EOSDIS Core System Project

Release B SDPS Data Management Subsystem Design Specification for the ECS Project

March 1996

Hughes Information Technology Systems
Upper Marlboro, MD

Release B SDPS Data Management Subsystem Design Specification for the ECS Project

March 1996

Prepared Under Contract NAS5-60000
CDRL Item #046

SUBMITTED BY

Rick Kochhar /s/ 3/20/96

Rick Kochhar, Release B CCB Chairman Date
EOSDIS Core System Project

Hughes Information Technology Systems
Upper Marlboro, Maryland

305-CD-023-002

This page intentionally left blank.

Preface

This document is one of eighteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release B of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-020, which provides an overview of the subsystems and 305-CD-039, the Data Dictionary, for those reviewing the object models in detail.

The SDPS and CSMS subsystem design specification documents for Release B of the ECS Project include:

305-CD-020	Release B Overview of the SDPS and CSMS Segment System Design Specification
305-CD-021	Release B SDPS Client Subsystem Design Specification
305-CD-022	Release B SDPS Interoperability Subsystem Design Specification
305-CD-023	Release B SDPS Data Management Subsystem Design Specification
305-CD-024	Release B SDPS Data Server Subsystem Design Specification
305-CD-025	Release B SDPS Ingest Subsystem Design Specification
305-CD-026	Release B SDPS Planning Subsystem Design Specification
305-CD-027	Release B SDPS Data Processing Subsystem Design Specification
305-CD-028	Release B CSMS Segment Communications Subsystem Design Specification
305-CD-029	Release B CSMS Segment Systems Management Subsystem Design Specification
305-CD-030	Release B GSFC Distributed Active Archive Center Design Specification
305-CD-031	Release B LaRC Distributed Active Archive Center Design Specification
305-CD-033	Release B EDC Distributed Active Archive Center Design Specification
305-CD-034	Release B ASF Data Center Distributed Active Archive Center Design Specification
305-CD-035	Release B NSIDC Distributed Active Archive Center Design Specification
305-CD-036	Release B JPL Distributed Active Archive Center Design Specification
305-CD-037	Release B ORNL Distributed Active Archive Center Design Specification
305-CD-038	Release B System Monitoring and Coordination Center Design Specification
305-CD-039	Release B Data Dictionary for Subsystem Design Specification

Object models presented in this document have been exported directly from CASE or DBMS tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (EDHS) at: URL http://edhs1.gsfc.nasa.gov.

This document is a formal contract deliverable with an approval code of 2; as such it requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once this document is approved, Contractor approved changes are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Drive
Upper Marlboro, MD 20774-5372

Abstract

This document presents the design of the Data Management Subsystem of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). It defines the Data Management Subsystem's Release B CSCI and HWCI structures, as well as subsystem design based on Level 4 requirements. The Release B Data Management Subsystem includes four CSCIs, the Data Dictionary Service, the Distributed Information Manager, the Local Information Manager, and the V0 Gateway.

Keywords: SDPS, management, CSCI, HWCI, Gateway, ODL, V0, client, DDICT, data, dictionary, DIMGR, distributed, information, manager, LIMGR, local

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Submitted as Final
iii through xiv	Submitted as Final
1-1 through 1-2	Submitted as Final
2-1 through 2-4	Submitted as Final
3-1 through 3-8	Submitted as Final
4-1 through 4-46	Submitted as Final
5-1 through 5-4	Submitted as Final
6-1 and 6-26	Submitted as Final
7-1 through 7-80	Submitted as Final
8-1 through 8-10	Submitted as Final
A-1 through A-6	Submitted as Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
305-CD-023-001	Preliminary	October 1995	95-0736
305-CD-023-002	Submitted as Final	March 1996	96-0225

This page intentionally left blank.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

This Release B SDPS Data Management Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) Item 046, with requirements specified in Data Item Description (DID) 305/DV2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management Segment (CSMS) and the Science and Data Processing Subsystem (SDPS) for Release B.

1.2 Scope

Release B provides support to EOS AM-1 Mission Operations and Science Operations, and it provides support to ESDIS Ground System Certification Testing for the EOS AM-1 and Landsat 7 missions. Release B provides services for the Landsat 7, COLOR, ADEOS II, SAGE III, DAO, TERS, ERS, RADARSAT, and ALT RADAR missions, and it provides product generation support for COLOR.

The Release B SDPS Data Management Subsystem Design Specification defines the progress of the design. It defines the Release B SDPS Data Management Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements.

This subsystem is on an incremental development track. It is released and reviewed in the form of Evaluation Packages (EP), and is therefore not part of the formal Release B Critical Design Review. The overview material for these components has been included in this document for information purposes only. Only the public interface classes and persistant database classes are defined in this document. Documentation requirements for incremental development track are described in the Systems Engineering Plan for the ECS Project (194-201-SEI-001)

This document reflects the February 14, 1996 Technical Baseline maintained by the contractor configuration control board in accordance with the ECS Technical Direction No. 11 dated December 6, 1994.

1.3 Document Organization

The document is organized to describe the Data Management Subsystem design as follows:

- Section 1 provides information regarding the identification, scope, status, and organization of this document.
- Section 2 provides a listing of related documents which were used as source information for this document.
- Section 3 provides an overview of the Subsystem, focusing on the high-level design concept. The section provides general background and context information for the subsystem.

- Section 4 contains the structure of the Data Dictionary computer software configuration item (CSCI) of the Data Management Subsystem.
- Section 5 contains the structure of the Local Information Manager CSCI of the Data Management Subsystem.
- Section 6 contains the structure of the Distributed Information Manager CSCI of the Data Management Subsystem.
- Section 7 contains the structure of the V0 Gateway CSCI of the Data Management Subsystem.

Section 8 contains the hardware configuration item (HWCI) design of the Data Management Subsystem. This includes the hardware design for Interoperability Subsystem.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

1.4 Status and Schedule

This submittal of DID 305/DV2 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. The submittal was reviewed during the Release B Incremental Design Review (IDR) and reflects changes to the design which resulted from that review.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which the scope and content of this Data Management Subsystem Design Specification is derived.

194-207-SE1-001	System Design Specification for the ECS Project

2.2 Applicable Documents

The following documents are referenced within this SDPS Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

209-CD-001-003	Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet
209-CD-002-003	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System
209-CD-003-003	Interface Control Document Between EOSDIS Core System (ECS) and EOS-AM Project for AM-1 Spacecraft Analysis Software
209-CD-004-003	Data Format Control Document for the Earth Observing System (EOS) AM-1 Project Data Base
209-CD-005-005	Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF)
209-CD-006-005	Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC)
209-CD-007-003	Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS)
209-CD-008-004	Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC)
209-CD-009-002	Interface Control Document Between EOSDIS Core System (ECS) and the Marshall Space Flight Center (MSFC) Distributed Active Archive Center (DAAC)
209-CD-011-004	Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System
305-CD-020-002	Overview of Release B SDPS and CSMS System Design Specification for the ECS Project
308-CD-001-005	Software Development Plan for the ECS Project

313-CD-006-002	Release B CSMS/SDPS Internal Interface Control Document for the ECS Project
515-CD-002-002	Release B Availability Models/Predictions for the ECS Project
516-CD-002-002	Release B Reliability Predictions for the ECS Project
518-CD-002-001	Release B Maintainability Predictions for the ECS Project
160-TP-004-001	User Pull Analysis Notebook [for the ECS Project]
IMSV0-PD-SD-002	Hughes STX, Messages and Development Data Dictionary for v4.5 IMS Client
423-41-03	Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document

2.3 Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify and clarify the information presented in this document. These documents are not binding on the content of the SDPS Design Specifications.

205-CD-002-002	Science User's Guide and Operations Procedure Handbook for the ECS Project. Part 4: Software Developer's Guide to Preparation, Delivery, Integration, and Test with ECS
206-CD-001-002	Version 0 Analysis Report for the ECS Project
209-CD-010-001	Interface Control Document Between EOSDIS Core System (ECS) and the Langley Research Center (LaRC) Distributed Active Archive Center (DAAC) Draft
302-CD-002-001	SDPS/CSMS Release A and FOS Release A and B Facilities Plan for the ECS Project
101-303-DV1-001	Individual Facility Requirements for the ECS Project, Preliminary
194-317-DV1-001	Prototyping and Studies Plan for the ECS Project
318-CD-000-XXX	Prototyping and Studies Progress Report for the ECS Project (monthly)
333-CD-003-002	SDP Toolkit Users Guide for the ECS Project
601-CD-001-004	Maintenance and Operations Management Plan for the ECS Project
604-CD-001-004	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002-003	Operations Concept for the ECS project: Part 2B -- ECS Release B,
604-CD-003-002	Operations Concept for the ECS Project: Part 2A -- ECS Release A
604-CD-004-001	Operations Concept for the ECS Project: Part 2 -- FOS
101-620-OP2-001	List of Recommended Maintenance Equipment for the ECS Project
194-703-PP1-001	System Design Review (SDR) Presentation Package for the ECS Project
194-813-SI4-002	Planning and Scheduling Prototype Results Report for the ECS Project
194-813-SI4-003	DADS Prototype One FSMS Product Operational Evaluation

194-813-SI4-004	DADS Prototype One STK Wolfcreek 9360 Automated Cartridge System Hardware Characterization Report
813-RD-009-001	DADS Prototype Two Multi-FSMS Product Integration Evaluation
828-RD-001-002	Government Furnished Property for the ECS Project
193-TP-626-001	GCDIS/UserDIS Study ECS Technical Paper, Draft 0.2
193-WP-118-001	Algorithm Integration and Test Issues for the ECS Project
193-WP-611-001	Science-based System Architecture Drivers for the ECS Project, Revision 1.0
193-WP-623-001	ECS Evolutionary Development White Paper
194-TP-266-002	Data Distribution Architecture Logical Object Model (LOM) for the ECS Project, Version 2.01
194-TP-267-001	Data Server Architecture Logical Object Model (LOM) for the ECS Project, Version 2.00
194-TP-313-001	ECS User Characterization Methodology and Results
194-TP-316-001	Data Compression Study for the ECS Project
194-TP-548-001	User Scenario Functional Analysis [for the ECS Project]
194-TP-569-001	PDPS Prototyping at ECS Science and Technology Laboratory, Progress Report #4
194-WP-901-002	EOSDIS Core System Science Information Architecture, White Paper, Working Paper
194-WP-902-002	ECS Science Requirements Summary, White Paper, Working Paper
194-WP-904-002	Multi-Track Development for the ECS Project, White Paper, Working Paper
194-WP-913-003	User Environment Definition for the ECS Project, White Paper, Working Paper
194-WP-914-001	CORBA Object Request Broker Survey for the ECS Project, White Paper, Working Paper
194-WP-918-001	DADS Prototype One FSMS Product Operational Evaluation, White Paper, Draft Report
194-WP-925-001	Science Software Integration and Test, White Paper, Working Paper
222-TP-003-008	Release Plan Content Description for the ECS Project
410-TD-001-002	ECS User Interface Style Guide Technical Data
420-WP-001-001	Maximizing the Use of COTS Software in the SDPS SDS Software Design, White Paper
430-TP-001-001	SDP Toolkit Implementation with Pathfinder SSM/I Precipitation Rate Algorithm, Technical Paper
423-16-01	Goddard Space Flight Center, Data Production Software and Science Computing Facility (SCF) Standards and Guidelines

423-16-02	Goddard Space Flight Center, PGS Toolkit Requirements Specification for the ECS Project
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System
540-022	Goddard Space Flight Center, Earth Observing System (EOS) Communications (Ecom) System Design Specification
560-EDOS-0211.0001	Goddard Space Flight Center, Interface Requirements Document Between EDOS and the EOS Ground System (EGS)

3. Release B SDPS Data Management Subsystem Overview

3.1 Subsystem Overview

The Release B SDPS Data Management Subsystem provides services which search for, locate and access data on behalf of a user or another program. Data management services decouple users and programs from the methods used by a site to access the data, and the manner in which the data have been named. The Release B SDPS Data Management Subsystem consists of four CSCIs, the Data Dictionary Service (DDICT), the Distributed Information Manager (DIMGR), and the V0 Gateway (GTWAY). The Data management hardware CI (DMGHW) provides the hardware support needed by Release B SDPS Data Management Subsystem and Interoperability subsystem software components.

3.2 Subsystem Structure

The Release B SDPS Data Management Subsystem is composed of four CSCIs and one HWCI:

- Data Dictionary Service (DDICT) is a software CI. It manages the definitions of data objects, attributes, domains (valid values), and access operations available via Science Data Servers (SDSRV), LIMGRs, GTWAYs, and DIMGRs. The DDICT information will be stored in a relational Database Management System (DBMS). The database will be replicated at each DAAC using the COTS DBMS software to perform the replication.
- Local Information Manager (LIMGR) is a software CI. It provides access to data and services at a site to the extent the underlying data servers make their data available via the LIMGR. The LIMGR accepts requests, such as a search, and produces and executes the corresponding requests that must occur at the data servers for that site. An operator must specify to the LIMGR what objects can be accessed at the various data servers at a site. The LIMGR can be modified by a site to provide site specific access to other data sources besides Data Servers.
- Distributed Information Manager (DIMGR) is a software CI. It provides access to data and services across sites. The DIMGR accepts requests, such as a search, and produces and executes the corresponding requests that must occur at the LIMGRs and/or data servers. An operator must specify which LIMGRs and data servers can be accessed by each DIMGR.
- V0 Gateway (GTWAY) is a software CI. It provides interoperability services between ECS data server and V0 client.
- Data Management HWCI (DMGHW) is a hardware component. It provides hardware for both Data Management subsystem and Interoperability subsystem CSCIs.

Figure 3.2-1 illustrates the Release B SDPS Data Management Subsystem context within ECS. For each of the context diagram flows, Table 3.2-1 contains a description of the flow.

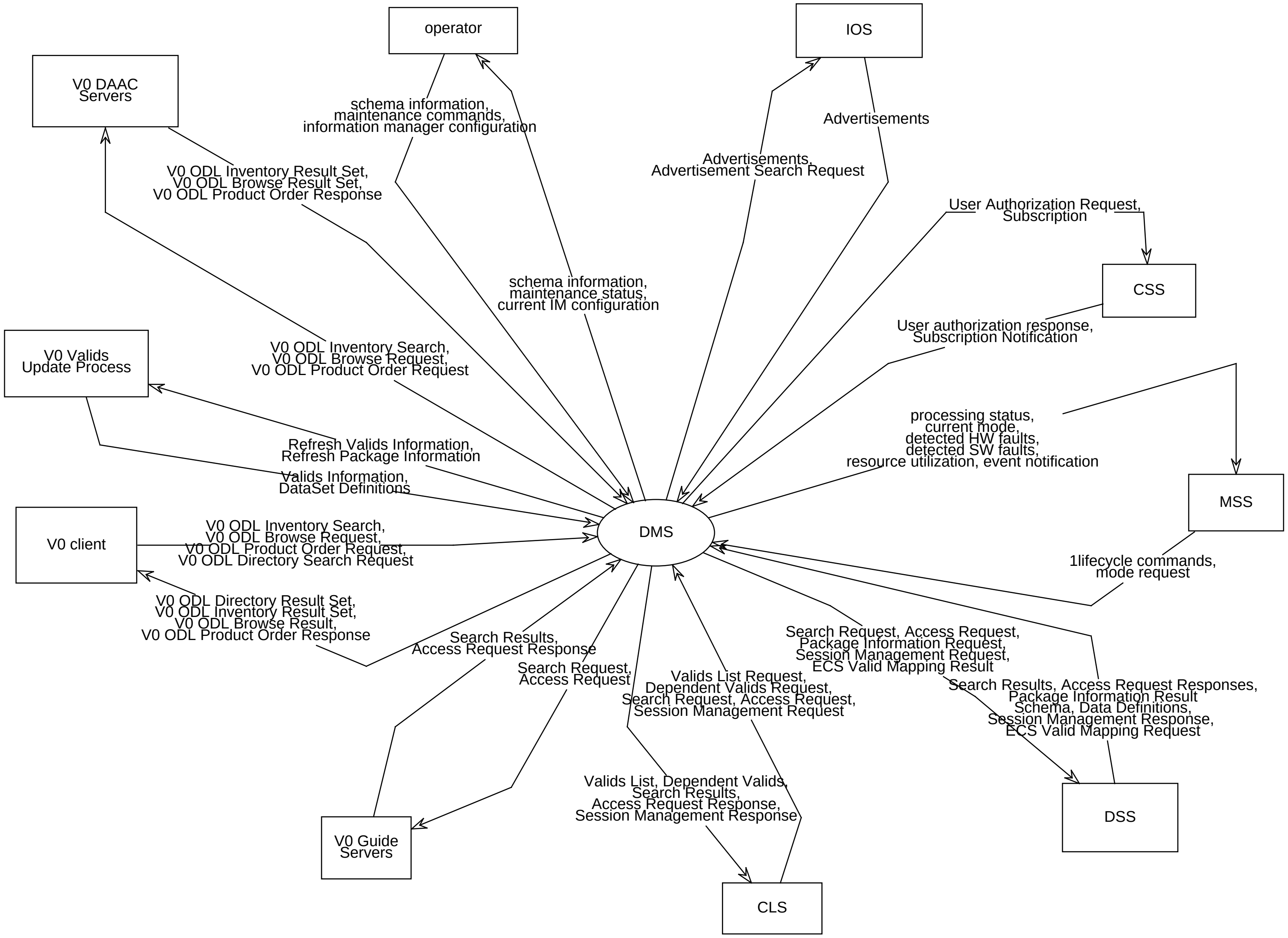

operator
IOS
V0 DAAC Servers
schema information, maintenance commands, information manager configuration
Advertisements
Advertisements, Advertisement Search Request
V0 ODL Inventory Result Set, V0 ODL Browse Result Set, V0 ODL Product Order Response
User Authorization Request, Subscription
CSS
schema information, maintenance status, current IM configuration
User authorization response, Subscription Notification
V0 Valids Update Process
V0 ODL Inventory Search, V0 ODL Browse Request, V0 ODL Product Order Request
processing status, current mode, detected HW faults, detected SW faults, resource utilization, event notification
Refresh Valids Information, Refresh Package Information
Valids Information, DataSet Definitions
MSS
DMS
V0 client
V0 ODL Inventory Search, V0 ODL Browse Request, V0 ODL Product Order Request, V0 ODL Directory Search Request
1lifecycle commands, mode request
V0 ODL Directory Result Set, V0 ODL Inventory Result Set, V0 ODL Browse Result, V0 ODL Product Order Response
Search Results, Access Request Response
Search Request, Access Request, Package Information Request, Session Management Request, ECS Valid Mapping Result
Search Request, Access Request
Valids List Request, Dependent Valids Request, Search Request, Access Request, Session Management Request
Search Results, Access Request Responses, Package Information Result Schema, Data Definitions, Session Management Response, ECS Valid Mapping Request
Valids List, Dependent Valids, Search Results, Access Request Response, Session Management Response
DSS
V0 Guide Servers
CLS

Figure 3.2-1. Data Management Subsystem Context

In the table, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column. The frequency information will be updated as the interfaces are fully defined. The shaded rows are flows new to Release B.

Table 3.2-1. Data Management Subsystem Interfaces (1 of 4)

Source	Destination	Data Types	Data Volume	Frequency
Client	Data Management	Valids List Request	low	as needed for initialization of the client interface
Data Management	Client	Valids List	medium	in response to request
Client	Data Management	Dependent Valids Request	low	as needed for initialization of dependencies in client interface
Data Management	Client	Dependent Valids	medium	in response to request
Client	Data Management	Search Request	low	as requested by user
Data Management	Client	Search Result	low-high	in response to request
Client	Data Management	Access Request	low	as requested by user
Data Management	Client	Access Request Response	low	in response to request
Client	Data Management	Session Management Request	low	as requested by user
Data Management	Client	Session Management Response	low	in response to request
V0 Client	Data Management	V0 Directory Search Requests	low	as requested
Data Management	V0 Client	V0 Directory Search Result Set	low	in response to request
V0 Client	Data Management	V0 Inventory Search Requests	low	as requested
Data Management	Data Server	Search Request	low	in response to a V0 inventory search request
Data Server	Data Management	Search Result	low-high	in response to ECS inventory search request
Data Management	V0 Client	V0 Inventory Result Set	low-high	in response to ECS inventory result Set
V0 Client	Data Management	V0 Browse Request	low	as requested
Data Management	Data Server	Access Request	low	in response to V0 Browse and Product Order Request

Table 3.2-1. Data Management Subsystem Interfaces (2 of 4)

Source	Destination	Data Types	Data Volume	Frequency
Data Server	Data Management	Access Request Response	low-medium	in response to ECS Browse and Product Order Request
Data Management	V0 Client	V0 Browse Result	low-medium	in response to ECS Browse Result
V0 Client	Data Management	V0 Product Order Request	low	frequency dependent on user input
Data Management	V0 Client	V0 Product Order Response	low	in response to ECS Product Request Response
Data Server	Data Management	Schema	low	whenever new ESDT is created or an update to the ESDT Descriptor is made
Data Server	Data Management	Data Definitions	low	in response to data definition request
Data Server	Data Management	ECS Valid Mapping Request	low	as requested
Data Management	Data Server	ECS Valid Mapping Result	low	in response to the ECS Valid Mapping request
Data Management	Data Server	Session Management Request	low	as needed to manage sessions
Data Server	Data Management	Session Management Response	low	in response to session management request
Interoperability	Data Management	Advertisements	low	as requested
Data Management	Interoperability	Advertisement Search Request	low	as needed to find data server services
Data Management	Data Server	Package Information Request	low	as requested by the administrator
Data Server	Data Management	Package Information Result	low	in response to the package information request
Data Management	V0 DAAC Servers	V0 ODL Inventory Search Request	low	in response to an ECS query of a V0 data set.
V0 DAAC Servers	Data Management	V0 ODL Inventory Result Set	low	in response to the V0 Inventory search
Data Management	V0 DAAC Server	V0 ODL Browse Request	low	in response to ECS client request of V0 browse data.
V0 DAAC Server	Data Management	V0 ODL Browse Result Set	low	in response to ECS request of V0 browse.
Data Management	V0 DAAC Server	V0 ODL Product Order Request	low	in response to ECS client request of V0 products

Table 3.2-1. Data Management Subsystem Interfaces (3 of 4)

Source	Destination	Data Types	Data Volume	Frequency
V0 DAAC Server	Data Management	V0 ODL Product Order Response	low	in response to ECS request for V0 product
operator	Data Management	schema information	low	as necessary to update LIM and DIM schemas
Data Management	operator	schema information	low	as necessary to view schema information
operator	Data Management	maintenance commands	low	as necessary for administration tasks
Data Management	operator	maintenance status	low	in response to a maintenance command.
operator	Data Management	information manager configuration	low	as necessary for administration tasks
Data Management	operator	current IM configuration	low	as needed when reconfiguring DIMs and LIMs
Data Management	CSS	user authorization request	low	as required for validation of user access.
CSS	Data Management	user authentication response	low	in response to authorization request
Data Management	CSS	subscription	low	as required for getting updates to DDICT information.
CSS	Data Management	subscription notification	low	in response to subscription event, upon update to DDICT information.
Data Management	MSS	processing status	low	as needed to show current status
Data Management	MSS	current mode	low	in response to mode request
MSS	Data Management	mode request	low	as needed to determine mode of operations
Data Management	MSS	detected HW faults	low	in response to HW error
Data Management	MSS	detected SW faults	low	in response to OS error
Data Management	MSS	resource utilization	low	as necessary to define use of resources
Data Management	MSS	event notification	low	in response to general events.
MSS	Data Management	lifecycle commands	low	as needed to startup or shutdown servers
Data Management	V0 Valids Update Process	Refresh Package Information	low	in response to the Package Information Result from data server

Table 3.2-1. Data Management Subsystem Interfaces (4 of 4)

Source	Destination	Data Types	Data Volume	Frequency
Data Management	V0 Valids Update Process	Refresh Valids Information	low	in response to Data Server Schema Export
V0 Valids Update Process	Data Management	Valids Information	low	as needed to update the data dictionary
V0 Valids Update Process	Data Management	DataSet Definitions	low	as needed to update data collection info in data dictionary.

3.3 Subsystem Design Rationale

The following are the drivers for the design of the Release B SDPS Data Management Subsystem:

a. Decouple users and programs from the methods used by sites to access the data, and the manner in which the data are named.

b. Provide interoperability between V0 client and ECS data server.

c. Simplify data administration

d. Provide site autonomy.

e. Provide interoperability between the ECS Workbench and the V0 IMS Servers.

f. Provide services for the ECS Workbench to execute distributed queries, that is queries that access multiple sites and possibly correlate results between sites.

g. Provide services for the ECS Workbench to execute queries that access multiple data servers within one site and possibly correlate results between data servers.

3.4 Subsystem Physical Design

The Data Management Subsystem hardware is described in section 8 of this document. Briefly, there is one HP K400 server that acts as the primary machine in support of the Data Management Subsystem. Each of the software CIs; DDICT, DIMGR, LIMGR, and GTWAY, is a Unix process on the server. In addition to these processes the DCE Client processes and Sybase server processes must be running. Figure 3.4-1 illustrates the current process to hardware mapping. The processes shown on the figure as follows:

- Log Transaction Manager - This is a Sybase process that reads the Sybase transaction log looking for transactions to send to the Replication Server.
- Sybase Replication Server - This is a Sybase process that receives replication requests from the Log Transaction Manager and sends SQL requests to other Sybase servers at other sites.
- Sybase SQL Server - This is the Sybase DBMS server process.
- DIM Server - This is the Distributed Information Manager with a global view of all the ECS data.
- LIM Server - This is the Local Information Manager process that provides access to all the data at the site.
- V0 GTWAY - This is the V0 Gateway process that provides two-way interoperability between ECS and V0.

- DDICT Server - This is the DDICT server that provides access to the Data Dictionary information at a site.
- Advertising Server - This is the Advertising Application Server that is a DCE server that applications search to find service and product advertisements. It belongs to the Interoperability Subsystem, but resides on the Data Management hardware.
- HTTP Server - This is the Advertising Navigator Server which is a WWW server that provides access to the Advertisements. The client subsystem also provides access to the DDICT information through this WWW Server.
- CGI Bin Programs - These are various programs from the Interoperability and Client Subsystem to search advertisements and data dictionary information using the WWW server.

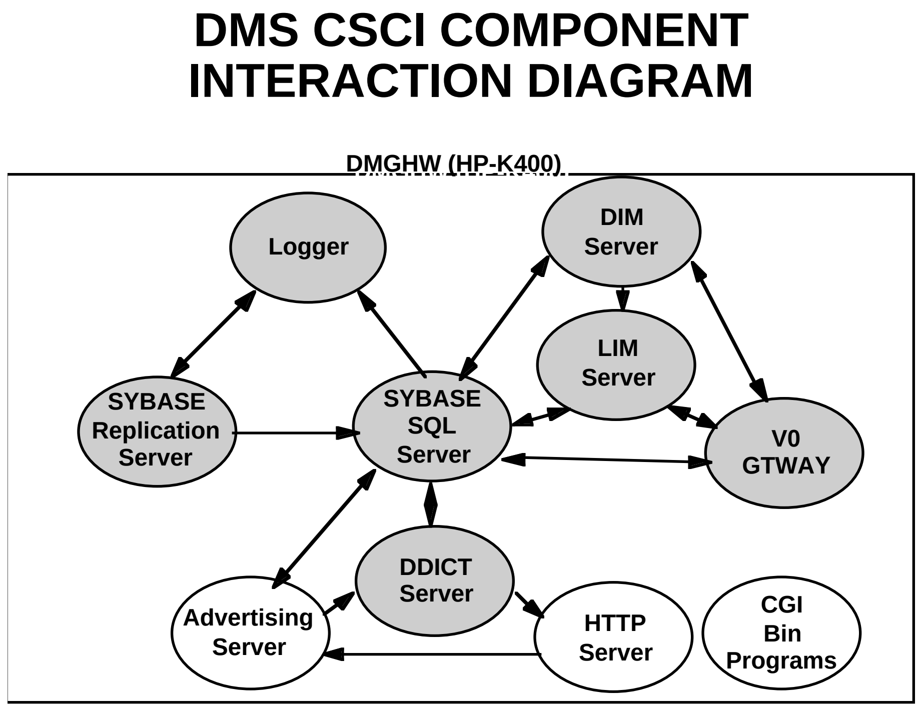


Figure 3.4-1 Data Management Physical Design

3.5 Key Mechanisms

The Data Management Subsystem is built using the following key mechanisms:

- Server Request Framework (SRF)
- Universal References (UR)
- Managed Process Framework (MPF)
- Event Logging
- Subscriptions

This page intentionally left blank.

4. DDICT - Data Dictionary Service CSCI

4.1 CSCI Overview

The DDICT provides access to databases containing information about data objects, their attributes, their operations, and the domains of the attributes. The DDICT describes the data objects accessible through data servers, LIMs, DIMs, and GTWAYs. The DDICT is used for informational support by users to retrieve definitions of the available items and as infrastructural support to the other CSCIs within the Data Management Subsystem (LIMGR, DIMGR, and GTWAY).

Since this CSCI is on the incremental development track, requirements, schedules, scenarios, issues and internal designs are documented in a Software Development File (SDF). The object models in this document are provided to show the public interface that will be used by formal development track components to the DDICT.

The object model for the persistent data stored in the DDICT database is provided for informational purposes only. The heart of the DDICT database design is the Data Collection objects described in SDPS Database Design and Database Schema Specifications for the ECS Project (311-CD-002-004). The DDICT stores part of the Data Collection object model. The other attributes related to collections and their granules are stored as values in the DDICT database describing just the domains and formats of those attributes.

4.2 Data Dictionary Service Context

Figure 4.2-1 shows the context diagram for DDICT. The flows are defined in Table 3.2-1.

4.3 Data Dictionary Service Object Model

Data Dictionary Client Library Model

The DDICT Client Library object model shown in Figure 4.3.-1 defines the objects used by clients of the DDICT service. The primary functions provided by the DDICT service client library are to provide search capabilities and the ability to add, modify, or delete schema information held in the DDICT. The DDICT Client Library is built on the Server Request Framework described in 305-CD-028-002. It provides two possible interfaces to the anyone wishing to submit requests to the DDICT service. The requests can be synchronous or asynchronous. The methods NewSearchRequest and NewSchemaRequest on the DmDdRequestServer, which is the session controller of the client library, provide the asynchronous method. They return objects of DmDdSearchRequest and DmDdSchemaRequest respectively. These two objects are used to submit an asynchronous request. The client will provide a function to these request objects in order for the DDICT Client library to inform the calling program of the progress of the search.

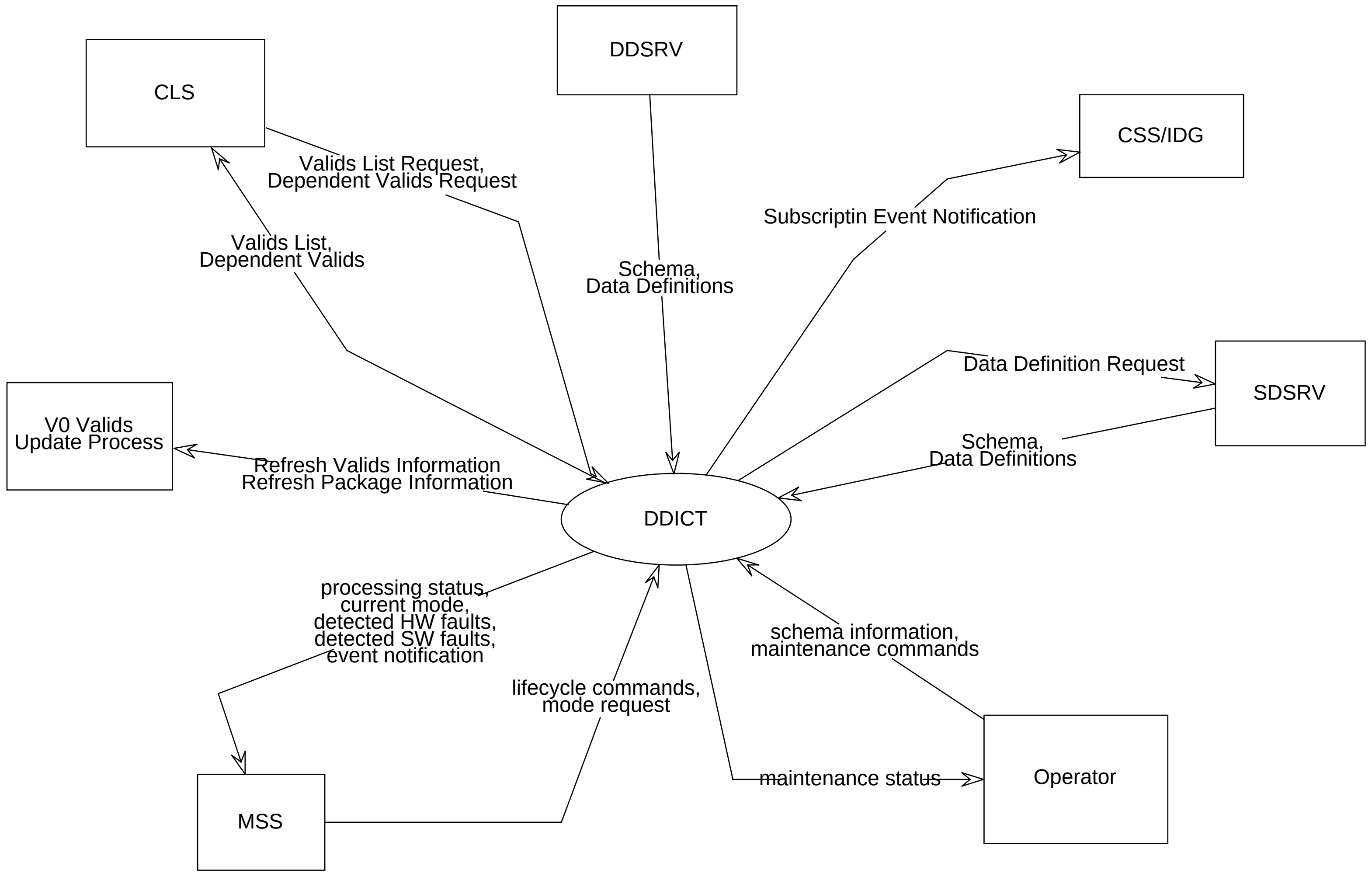


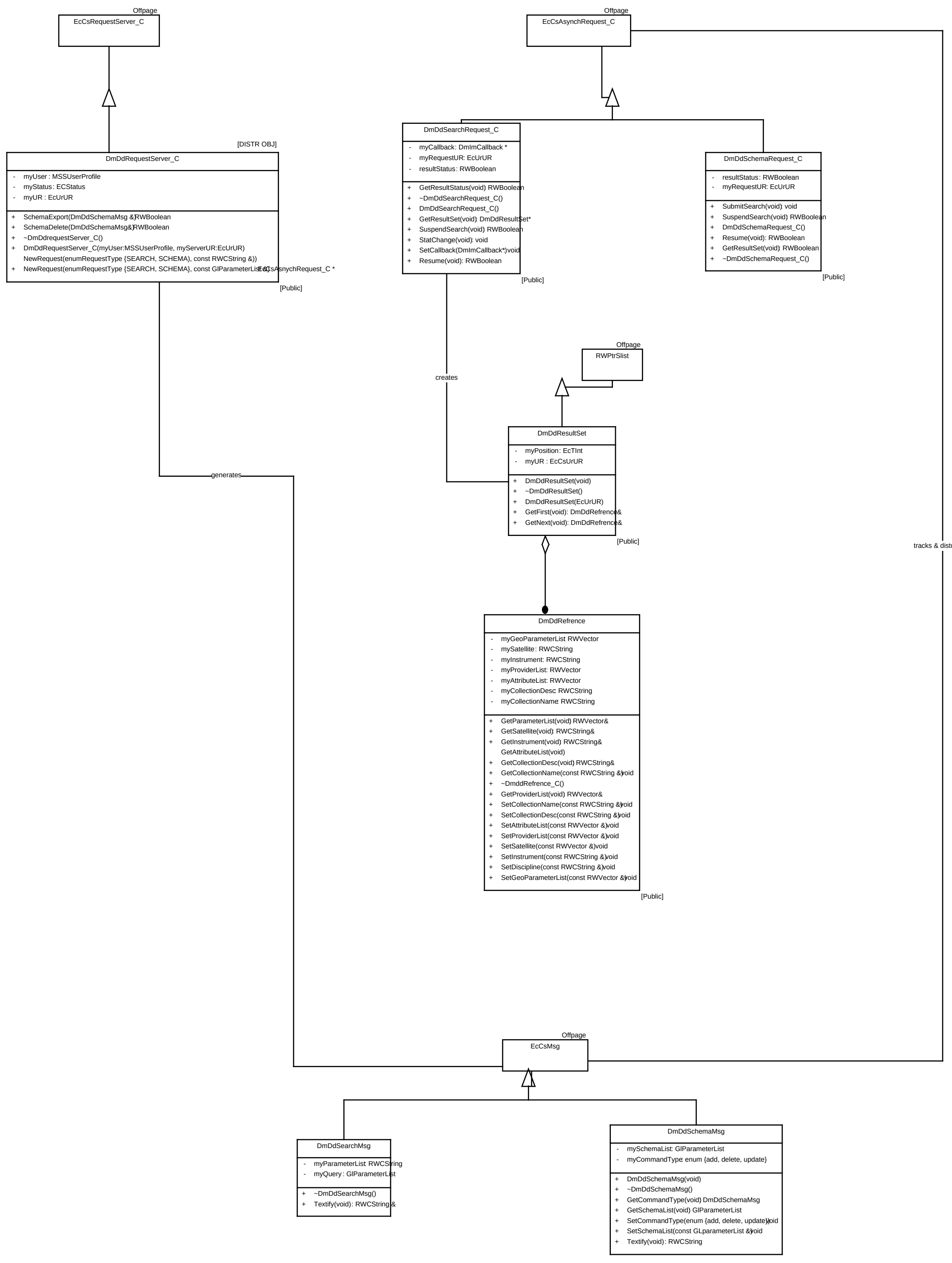
Figure 4.2-1. DDICT Context Diagram

The synchronous communications are provided by the other operations on the DmDdRequestServer object. For example, the SchemaExport operation will take a schema request and synchronously execute it against the server. The client thread will be waiting until the SchemaExport operation completes. This type of interaction with the DDICT service is useful when the caller must obtain an error or success code before going on to the next operation.

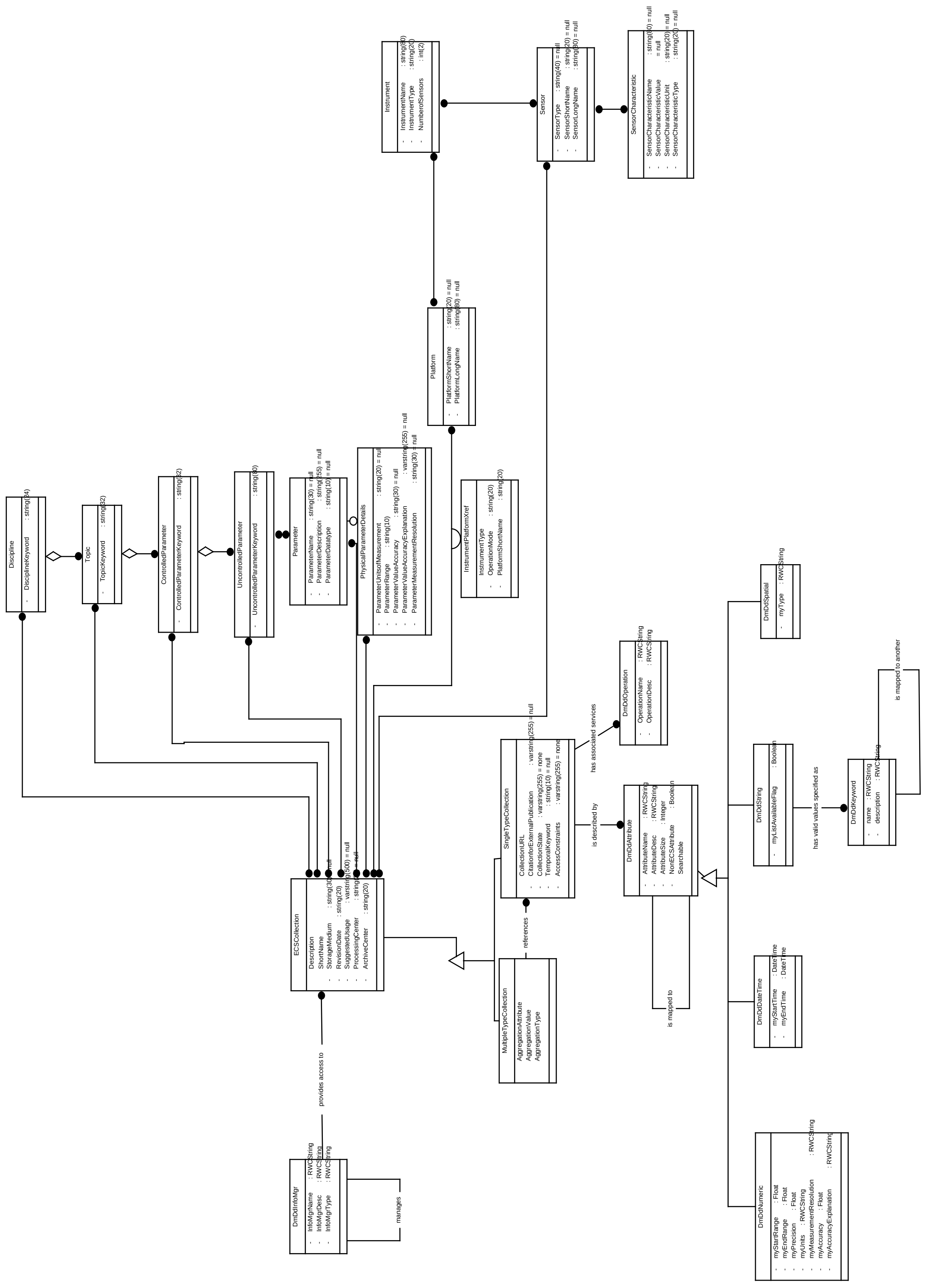
Data Dictionary Service Persistent Model

The Data Dictionary Service Persistent Model (shown in Figure 4.3.-2) defines the data stored in the DDICT database to support the DDICT service functionality. The central concept behind the model is the ECSCollection concept defined in the SDPS Database Design and Database Schema Specifications for the ECS Project (311-CD-002-003). This document defines a collection as a site specific grouping of objects. The objects may be other data collections or granule objects. In the DDICT model, all the objects are related to the collection level material. There are no granule level objects in the DDICT service database, although there are attributes described about the granule level metadata.

The ECSCollection and its subclasses object are related to a group of objects that define the characteristics of the collection, for example, the temporal and spatial location of the collection, the instrument, discipline, and geophysical parameters. The Attribute class defines the attributes that are down at the granule level that describe the collection. Those attributes may be metadata attributes defined in a DBMS or some attribute within the data itself (evolutionary requirement). At the DDICT service, the physical storage mechanism is not relevant. The DmDdInfoMgrs class defines the information managers in the ECS federation that access these collections. This is the key to the parsing of queries and routing subqueries to lower level ECS components, for example, from a DIMGR to a LIMGR. Collections can have multiple collections related to them. This allows a LIMGR to aggregate the multiple SDSRV collections into one advertised collection for the site.


Offpage
EcCsRequestServer_C
Offpage
EcCsAsynchRequest_C
[DISTR OBJ]
DmDdRequestServer_C
- myUser : MSSUserProfile
- myStatus : ECStatus
- myUR : EcUrUR
+ SchemaExport(DmDdSchemaMsg &) RWBoolean
+ SchemaDelete(DmDdSchemaMsg&) RWBoolean
+ ~DmDdrequestServer_C()
+ DmDdRequestServer_C(myUser:MSSUserProfile, myServerUR:EcUrUR)
NewRequest(enumRequestType {SEARCH, SCHEMA), const RWCString &))
+ NewRequest(enumRequestType {SEARCH, SCHEMA}, const GlParameterList &) EcCsAsnychRequest_C *
[Public]
DmDdSearchRequest_C
- myCallback: DmImCallback *
- myRequestUR: EcUrUR
- resultStatus: RWBoolean
+ GetResultStatus(void) RWBoolean
+ ~DmDdSearchRequest_C()
+ DmDdSearchRequest_C()
+ GetResultSet(void): DmDdResultSet*
+ SuspendSearch(void) RWBoolean
+ StatChange(void): void
+ SetCallback(DmImCallback*) void
+ Resume(void): RWBoolean
[Public]
DmDdSchemaRequest_C
- resultStatus: RWBoolean
- myRequestUR: EcUrUR
+ SubmitSearch(void): void
+ SuspendSearch(void) RWBoolean
+ DmDdSchemaRequest_C()
+ Resume(void): RWBoolean
+ GetResultSet(void): RWBoolean
+ ~DmDdSchemaRequest_C()
[Public]
creates
Offpage
RWPtrSlist
DmDdResultSet
- myPosition: EcTInt
- myUR : EcCsUrUR
+ DmDdResultSet(void)
+ ~DmDdResultSet()
+ DmDdResultSet(EcUrUR)
+ GetFirst(void): DmDdRefrence&
+ GetNext(void): DmDdRefrence&
[Public]
generates
tracks & distributes
DmDdRefrence
- myGeoParameterList RWVector
- mySatellite: RWCString
- myInstrument: RWCString
- myProviderList: RWVector
- myAttributeList: RWVector
- myCollectionDesc RWCString
- myCollectionName RWCString
+ GetParameterList(void) RWVector&
+ GetSatellite(void): RWCString&
+ GetInstrument(void) RWCString&
GetAttributeList(void)
+ GetCollectionDesc(void) RWCString&
+ GetCollectionName(const RWCString &) void
+ ~DmddRefrence_C()
+ GetProviderList(void) RWVector&
+ SetCollectionName(const RWCString &) void
+ SetCollectionDesc(const RWCString &) void
+ SetAttributeList(const RWVector &) void
+ SetProviderList(const RWVector &) void
+ SetSatellite(const RWVector &)void
+ SetInstrument(const RWCString &) void
+ SetDiscipline(const RWCString &) void
+ SetGeoParameterList(const RWVector &) void
[Public]
Offpage
EcCsMsg
DmDdSearchMsg
- myParameterList: RWCString
- myQuery : GlParameterList
+ ~DmDdSearchMsg()
+ Textify(void): RWCString &
DmDdSchemaMsg
- mySchemaList: GlParameterList
- myCommandType: enum {add, delete, update}
+ DmDdSchemaMsg(void)
+ ~DmDdSchemaMsg()
+ GetCommandType(void) DmDdSchemaMsg
+ GetSchemaList(void) GlParameterList
+ SetCommandType(enum {add, delete, update}) void
+ SetSchemaList(const GLparameterList &) void
+ Textify(void): RWCString

4.3-1. DDICTClient Object Model Diagram



Discipline
DisciplineKeyword : string(24)
Topic
TopicKeyword : string(32)
ControlledParameter
ControlledParameterKeyword : string(32)
UncontrolledParameter
UncontrolledParameterKeyword : string(40)
Parameter
ParameterName : string(30) = null
ParameterDescription : string(255) = null
ParameterDatatype : string(10) = null
PhysicalParameterDetails
ParameterUnitsofMeasurement : string(20) = null
ParameterRange : string(10)
ParameterValueAccuracy : string(30) = null
ParameterValueAccuracyExplanation : varstring(255) = null
ParameterMeasurementResolution : string(30) = null
InstrumentPlatformXref
InstrumentType : string(20)
OperationMode
PlatformShortName : string(20)
Platform
PlatformShortName : string(20) = null
PlatformLongName : string(80) = null
Instrument
InstrumentName : string(80)
InstrumentType : string(20)
NumberofSensors : int(2)
Sensor
SensorType : string(40) = null
SensorShortName : string(20) = null
SensorLongName : string(80) = null
SensorCharacteristic
SensorCharacteristicName : string(80) = null
SensorCharacteristicValue = null
SensorCharacteristicUnit : string(20) = null
SensorCharacteristicType : string(20) = null
ECSCollection
Description
ShortName : string(30) = null
StorageMedium : string(20)
RevisionDate
SuggestedUsage : varstring(500) = null
ProcessingCenter : string(20) = null
ArchiveCenter : string(20)
SingleTypeCollection
CollectionURL : varstring(255) = null
CitationforExternalPublication : varstring(255) = none
CollectionState : string(10) = null
TemporalKeyword
AccessConstraints : varstring(255) = none
MultipleTypeCollection
AggregationAttribute
AggregationValue
AggregationType
references
provides access to
DmDdInfoMgr
InfoMgrName : RWCString
InfoMgrDesc : RWCString
InfoMgrType : RWCString
manages
has associated services
DmDdOperation
OperationName : RWCString
OperationDesc : RWCString
is described by
DmDdAttribute
AttributeName : RWCString
AttributeDesc : RWCString
AttributeSize : Integer
NonECSAttribute : Boolean
Searchable
is mapped to
DmDdSpatial
myType : RWCString
DmDdString
myListAvailableFlag : Boolean
has valid values specified as
DmDdKeyword
name : RWCString
description : RWCString
is mapped to another
DmDdDateTime
myStartTime : DateTime
myEndTime : DateTime
DmDdNumeric
myStartRange : Float
myEndRange : Float
myPrecision : Float
myUnits : RWCString
myMeasurementResolution : RWCString
myAccuracy : Float
myAccuracyExplanation : RWCString

4-3-2. DDICTPersistentB Object Model Diagram

4.3.1 ControlledParameter Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:

Attributes:

ControlledParameterKeyword -
Data Type: string(32)
Privilege: Private
Default Value:
Contraints:not null
Non Persisent Flag:False

Operations:

None

Associations:

The ControlledParameter class has associations with the following classes:
Class: ECSCollection
Topic (Aggregation)

4.3.2 Discipline Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This class provides the highest level (of 5) levels of description of the collection content. GCMD keywords are used to describe the general discipline area of the collection. A collection can conceivably cover several disciplines.

Attributes:

DisciplineKeyword -
Data Type: string(24)
Privilege: Private
Default Value:
Contraints:not null
Non Persisent Flag:False

Operations:

None

Associations:

The Discipline class has associations with the following classes:
Class: ECSCollection

4.3.3 DmDdAttribute Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class defines attributes that relate to collections. All of the attributes defined in the SDPS Database Design and Database Schema Specifications for the ECS Project (311-CD-002-003) will be contained in this class. Other attributes can be specified but will be flagged as "non-ECS". This allows for external data providers to have other attributes or attributes with different definitions than those defined in the 311 Specification.

Attributes:

AttributeDesc - A description of the attribute. This provides the user with a description of the content of this attribute when populated.
Data Type: RWCString
Privilege: Private
Default Value:

AttributeName - The name of the attribute. The names for ECS baseline attributes, must be unique. Other organizations may redefine ECS attributes by using a different name.

Data Type:	RWCString
Privilege:	Private
Default Value:	

AttributeSize - The size of the attribute.

Data Type:	Integer
Privilege:	Private
Default Value:	

NonECSAttribute - This boolean flag defines whether the attribute is in the ECS baseline model or not.

Data Type:	Boolean
Privilege:	Private
Default Value:	

Searchable

Operations:

None

Associations:

The DmDdAttribute class has associations with the following classes:
Class: SingleTypeCollection isdescribedby
Class: DmDdAttribute ismappedto

4.3.4 DmDdDateTime Class

Parent Class:	DmDdAttribute
Public:	No
Distributed Object:	No
Persistent Class:	

Purpose and Description:
This class defines the domain of date/time attributes. It gives the range of valid date/time values.

Attributes:

myEndTime - This is the latest date/time value of the attribute.
Data Type: DateTime
Privilege: Private
Default Value:

myStartTime - This is the earliest date/time value of the attribute.
Data Type: DateTime
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The DmDdDateTime class has associations with the following classes:
None

4.3.5 DmDdInfoMgr Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class defines the information managers in the system. This includes DIMGRs, LIMGRs, SDSRVs, and DDSRVs. This class is related to the collections to show which information managers can access which collections.

Attributes:

InfoMgrDesc - Describes the purpose of the information manager. For example, a description of the general data that is accessed by a particular DIMGR.
Data Type: RWCString
Privilege: Private
Default Value:

InfoMgrName - The name of the information manager. This is used when looking up the information manager to actually connect to the services.
Data Type: RWCString
Privilege: Private
Default Value:

InfoMgrType - This specifies the type of the information manager which will be either DIMGR, LIMGR, GTWAY, SDSRV, or DDSRV.
Data Type: RWCString
Privilege: Private
Default Value:

Operations:

None

Associations:

The DmDdInfoMgr class has associations with the following classes:
Class: DmDdInfoMgr manages
Class: ECSCollection providesaccessto

4.3.6 DmDdKeyword Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class holds valid values for attributes of string type.

Attributes:

description - Describes the valid value. This might be a simple expansion of the valid value or a description of the meaning of the value or both.
Data Type: RWCString
Privilege: Private
Default Value:

name - The value of the keyword. For example a keyword value associated with a satellite might be NOAA-9.

Data Type:	RWCString
Privilege:	Private
Default Value:	

Operations:

None

Associations:

The DmDdKeyword class has associations with the following classes:
Class: DmDdString hasvalidvaluesspecifiedas
Class: DmDdKeyword ismappedtoanother

4.3.7 DmDdNumeric Class

Parent Class:	DmDdAttribute
Public:	No
Distributed Object:	No
Persistent Class:	

Purpose and Description:
This class defines the characteristics of a numeric attribute.

Attributes:

myAccuracy - The accuracy of the value. For example, 0.01 would be accurate to the second decimal place.

Data Type:	Float
Privilege:	Private
Default Value:	

myAccuracyExplanation - An explanation of the accuracy value.

Data Type:	RWCString
Privilege:	Private
Default Value:	

myEndRange - The highest value of possible domain values.

Data Type:	Float
Privilege:	Private
Default Value:	

myMeasurementResolution - The resolution of the numeric attribute.

Data Type:	RWCString
Privilege:	Private
Default Value:	

myPrecision - The precision of the numeric attribute, meaning the significant decimal places.

Data Type:	Float
Privilege:	Private
Default Value:	

myStartRange - The lowest value of possible domains.

Data Type:	Float
Privilege:	Private
Default Value:	

myUnits - The units of measurement for the numeric value (if applicable).

Data Type:	RWCString
Privilege:	Private
Default Value:	

Operations:

All Operations inherited from parent class

Associations:

The DmDdNumeric class has associations with the following classes:

None

4.3.8 DmDdOperation Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class defines the operations (or services) that can be performed on the collections. This class only gives the descriptions of the operations. For the full metadata about services, the advertising service must be consulted.

Attributes:

OperationDesc - The description of the operation. This briefly describes what the operation does.
Data Type: RWCString
Privilege: Private
Default Value:

OperationName - The name of the operation (or service). For example, the name might be Subset.
Data Type: RWCString
Privilege: Private
Default Value:

Operations:

None

Associations:

The DmDdOperation class has associations with the following classes:
Class: SingleTypeCollection hasassociatedservices

4.3.9 DmDdRefrence Class

Parent Class: Not Applicable
Public: Yes
Distributed Object:No
Purpose and Description:
From this object the calling object can get all the available information about a collection. The attribute list provides information for core and noncore attributes, including domain values and data types.

Attributes:

myAttributeList - the list of attributes available for search and access for this
collection. This list will be a list of attribute classes from which the calling object can find out everything about an attribute.
Data Type: RWVector
Privilege: Private
Default Value:

myCollectionDesc - The description of the data collection. This is a short summary description rather than the longer directory level description.
Data Type: RWCString
Privilege: Private
Default Value:

myCollectionName - The name of the data collection.
Data Type: RWCString
Privilege: Private
Default Value:

myGeoParameterList - The list of geophysical parameters available in the data collection. This list will contain a parameter class which will provide all the information necessary about the geophysical parameters.
Data Type: RWVector
Privilege: Private
Default Value:

myInstrument - The instrument that was used for the data collection. Data Type: RWVector
Data Type: RWCString
Privilege: Private
Default Value:

myProviderList - The list of providers (SDSRVs, LIMs, DIMS) that can search and access the data collection.
Data Type: RWVector
Privilege: Private
Default Value:

mySatellite - The satellite that the instrument resided on.
Data Type: RWCString
Privilege: Private
Default Value:

Operations:

GetAttributeList - Returns the list of attributes associated with the collection. This includes core as well as product specific attribtues.
Arguments: void

GetCollectionDesc - Returns the list of myCollectionDesc attribute.
Arguments: void
Return Type: RWCString&
Privilege: Public

GetCollectionName - Returns the name of the collection specified in the result.
Arguments: const RWCString &
Return Type: void
Privilege: Public

GetInstrument - Returns myInstrument attribute.
Arguments: void
Return Type: RWCString&
Privilege: Public

GetParameterList - Returns the list go geophysical parameters that are contained in the data collection.
Arguments: void
Return Type: RWVector&
Privilege: Public

GetProviderList - Returns the list of provider that supply the collection. This provider list includes DIMGRs and SDSRVs.
Arguments: void
Return Type: RWVector&
Privilege: Public

GetSatellite - Returns the list of satellites available.

Arguments:	void
Return Type:	RWCString&
Privilege:	Public

SetAttributeList - Sets the myAttributeList attribute to the value of the argument.

Arguments:	const RWVector &
Return Type:	void
Privilege:	Public

SetCollectionDesc - Sets the myCollectionDesc attribute to the value of the argument.

Arguments:	const RWCString &
Return Type:	void
Privilege:	Public

SetCollectionName - Sets the myCollectionName attribute to the value specified in the argument.

Arguments:	const RWCString &
Return Type:	void
Privilege:	Public

SetDiscipline - Set the mydiscipline attribute to the value of the argument.

Arguments:	const RWCString &
Return Type:	void
Privilege:	Public

SetGeoParameterList - Set the mygeoParameterList attribute to the value of the argument.

Arguments:	const RWVector &
Return Type:	void
Privilege:	Public

SetInstrument - Set the myInstrument attribute to the value of the argument.

Arguments:	const RWCString &
Return Type:	void
Privilege:	Public

SetProviderList - Set the myProviderList attribute to the value of the argument.

Arguments:	const RWVector &
Return Type:	void
Privilege:	Public

SetSatellite - Sets the mySatellite attribute to the value of the argument.

Arguments:	const RWVector &
Return Type:	void
Privilege:	Public

~DmddRefrence_C - A destructor.

Arguments:	
Return Type:	Void
Privilege:	Public

Associations:

The DmDdRefrence class has associations with the following classes:
DmDdResultSet (Aggregation)

4.3.10 DmDdRequestServer_C Class

Parent Class:	EcCsRequestServer_C
Public:	Yes
Distributed Object:	Yes

Purpose and Description:
This class is inheriting from EcCsRequestServer_C. It is used to manage the user session and to create objects capable of communicationg asynch between a client and a server.

Attributes:

myStatus - The status of the request.

Data Type:	ECStatus
Privilege:	Private
Default Value:	

myUR - The universal reference to the request on the server side.

Data Type:	EcUrUR
Privilege:	Private
Default Value:	

myUser - This is the user profile of the user who created the request. It is used by the server to provide authorization to services and resources.
Data Type: MSSUserProfile
Privilege: Private
Default Value:

Operations:

DmDdRequestServer_C - Constructor called by a client application. User's profile is passed as a parameter.
Arguments: myUser:MSSUserProfile, myServerUR:EcUrUR
Return Type: Void
Privilege: Public
PDL: No PDL

NewRequest - This function is called by the user. The user provides the request type. Depending upon the request type, the object DmDdSearchRequest_C or DmDdSchemaRequest_C is created.
Arguments: enumRequestType {SEARCH, SCHEMA

NewRequest - This function is called by the user. Depending upon the request type, the object DmDdSearchRequest_C or DmDdSchemaRequest_C is created.
Arguments: enumRequestType {SEARCH, SCHEMA}, const GlParameterList &
Return Type: EcCsAsnychRequest_C *
Privilege: Public

SchemaDelete - This operation creates an object of class DmDdSchemaMsg.
Arguments: DmDdSchemaMsg&
Return Type: RWBoolean
Privilege: Public
PDL: No PDL

SchemaExport - This operation creates an object of class DmDdSchemaMsg.
Arguments: DmDdSchemaMsg &
Return Type: RWBoolean
Privilege: Public
PDL: No PDL

~DmDdrequestServer_C - A destructor.
Arguments:
Return Type: Void
Privilege: Public
PDL: No PDL

Associations:

The DmDdRequestServer_C class has associations with the following classes:
Class: EcCsMsg generates

4.3.11 DmDdResultSet Class

Parent Class: RWPtrSlist
Public: Yes
Distributed Object:No
Purpose and Description:
The object of this class is created by DmDdSearchRequest_C object to store set of results retrieved from the server.

Attributes:

myPosition - The current position of the pointer in the result set.
Data Type: EcTInt
Privilege: Private
Default Value:

myUR - The universal reference to this session.
Data Type: EcCsUrUR
Privilege: Private
Default Value:

Operations:

DmDdResultSet - A constructor to create an empty object.

Arguments:	void
Return Type:	Void
Privilege:	Public

DmDdResultSet - A constructor to create a result set given the UR to a previously saved result set.

Arguments:	EcUrUR
Return Type:	Void
Privilege:	Public

GetFirst - Returns pointer to the first dictionary reference in the result set.

Arguments:	void
Return Type:	DmDdRefrence&
Privilege:	Public

GetNext - Returns a pointer to the next dictionary refrence in the result set.

Arguments:	void
Return Type:	DmDdRefrence&
Privilege:	Public

~DmDdResultSet - A destructor to destroy an object.

Arguments:	
Return Type:	Void
Privilege:	Public

Associations:

The DmDdResultSet class has associations with the following classes:

Class: DmDdSearchRequest_C creates

4.3.12 DmDdSchemaMsg Class

Parent Class: EcCsMsg
Public: No
Distributed Object:No
Purpose and Description:
This class is a specialized command/message class that allows for insertion and update of schema information to the DDICT. The object of this class is created by the method od DmDdRequestServer_C.

Attributes:

myCommandType - The type of schema command, which is either update, delete, or add. This defines how to interpret the list of attributes.
Data Type: enum {add, delete, update}
Privilege: Private
Default Value:

mySchemaList - This is the list of schema updates or insertions. This list contains all the attributes, descriptions, and values needed to describe the schema.
Data Type: GlParameterList
Privilege: Private
Default Value:

Operations:

DmDdSchemaMsg - Consructor for a an empty schema message.
Arguments: void
Return Type: Void
Privilege: Public

GetCommandType - Returns the myCommandType attribute.
Arguments: void
Return Type: DmDdSchemaMsg
Privilege: Public

GetSchemaList - Returns the mySchemaList attribute.
Arguments: void
Return Type: GlParameterList
Privilege: Public

SetCommandType - Sets the command type attribute to the value of the argument.
Arguments: enum {add, delete, update}
Return Type: void
Privilege: Public

SetSchemaList - Sets the mySchemaList attribute to the value of the argument.
Arguments: const GLparameterList &
Return Type: void
Privilege: Public

Textify - Converts the command into a human readable format.
Arguments: void
Return Type: RWCString
Privilege: Public

~DmDdSchemaMsg - A destructor.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmDdSchemaMsg class has associations with the following classes:
None

4.3.13 DmDdSchemaRequest_C Class

Parent Class: EcCsAsynchRequest_C
Public: Yes
Distributed Object:No
Purpose and Description:
This class is inheriting from EcCsAsynchRequest_C. It creates an asynchronous object for each schema request. The object is created by EcsMsgHandler. At the end of processing this object is notified by the EcsMsgHandler. Upon notification, this object creates DmDdResultSet object to get and store result.

Attributes:

myRequestUR - my server request UR.
Data Type: EcUrUR
Privilege: Private
Default Value:

resultStatus - Status of the result. Set to TRUE when result is received.
Data Type: RWBoolean
Privilege: Private
Default Value:

Operations:

DmDdSchemaRequest_C - The constructor for an empty schema request object.
Arguments:
Return Type: Void
Privilege: Public

GetResultSet - This method creates DmDdResultSet object to retrieve and store results.
Arguments: void
Return Type: RWBoolean
Privilege: Public

Resume - This method resumes request submitted for a schema search to the DDICT server.
Arguments: void
Return Type: RWBoolean
Privilege: Public

SubmitSearch - This method suspends request submitted for a schema search to the DDICT server.
Arguments: void
Return Type: void
Privilege: Public

SuspendSearch - This method suspends request submitted for a schema search to the DDICT server.
Arguments: void
Return Type: RWBoolean
Privilege: Public

~DmDdSchemaRequest_C - A destructor.

Arguments:	
Return Type:	Void
Privilege:	Public

Associations:

The DmDdSchemaRequest_C class has associations with the following classes:

None

4.3.14 DmDdSearchMsg Class

Parent Class: EcCsMsg
Public: No
Distributed Object:No
Purpose and Description:
The object of this class is for a generic search message that can be submitted as a request to an ECS server. The object is created by the method of DmDdRequestServer_C.

Attributes:

myParameterList - The list of attributes to be queried.

Data Type:	RWCString
Privilege:	Private
Default Value:	

myQuery - The query in Earth Science Query Language.

Data Type:	GlParameterList
Privilege:	Private
Default Value:	

Operations:

Textify - Converts the command/message into a human readable format.

Arguments:	void
Return Type:	RWCString &
Privilege:	Public

~DmDdSearchMsg - A destructor.

Arguments:

Return Type:	Void
Privilege:	Public

Associations:

The DmDdSearchMsg class has associations with the following classes:

None

4.3.15 DmDdSearchRequest_C Class

Parent Class:	EcCsAsynchRequest_C
Public:	Yes
Distributed Object:	No

Purpose and Description:

This class is inheriting from EcCsAsynchRequest_C. This object is constructed via callback by the EcCsMsgHandler. It is notified by the EcCsMsgHandler about the status of the request at the server end as well as at the end of processing. Upon completion of processing, this object creates DmDdResultSet object to store result.

Attributes:

myCallback - The callback to notify an object about the result status.

Data Type:	DmImCallback *
Privilege:	Private
Default Value:	

myRequestUR - Server request UR.

Data Type:	EcUrUR
Privilege:	Private
Default Value:	

resultStatus - Status of the result. Set to TRUE when result is received. .

Data Type:	RWBoolean
Privilege:	Private
Default Value:	

Operations:

DmDdSearchRequest_C - A constructor called by DmDdRequestServer_C object to create an asych object.
Arguments:
Return Type: Void
Privilege: Public

GetResultSet - This method creates DmDdResultSet object to store results received from the DDICT server.
Arguments: void
Return Type: DmDdResultSet*
Privilege: Public

GetResultStatus - Status of the result. Set to TRUE when result is received.
Arguments: void
Return Type: RWBoolean
Privilege: Public

Resume - This method resumes request submitted for a schema search to the DDICT server.
Arguments: void
Return Type: RWBoolean
Privilege: Public

SetCallback - Sets myCallback attribute.
Arguments: DmImCallback*
Return Type: void
Privilege: Public

StatChange - This operation indicates change of status of the result received.
Arguments: void
Return Type: void
Privilege: Public

SuspendSearch - This method suspends request submitted to the server.
Arguments: void
Return Type: RWBoolean
Privilege: Public

~DmDdSearchRequest_C - A destructor.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmDdSearchRequest_C class has associations with the following classes:
Class: DmDdResultSet creates

4.3.16 DmDdSpatial Class

Parent Class: DmDdAttribute
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class defines the characteristics of spatial attributes. It will reference types defined in the SDPS Database Design and Database Schema Specifications for the ECS Project (311-CD-002-003).

Attributes:

myType - This is the type of spatial representation. In the Locality model of the SDPS Database Design and Database Schema Specifications for the ECS Project (311-CD-002-003) there are many spatial representations. This attribute will identify which of these representations, the attribute can be defined by.
Data Type: RWCString
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The DmDdSpatial class has associations with the following classes:
None

4.3.17 DmDdString Class

Parent Class: DmDdAttribute
Public: No
Distributed Object: No
Persistent Class:
Purpose and Description:
This class defines attributes of string type. It's only purpose is to relate valid values to the DmDdKeyword class.

Attributes:

myListAvailableFlag - This boolean flag defines whether there are valid value keywords associated with this attribute. Some string attributes like descriptions will not have valid values.
Data Type: Boolean
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The DmDdString class has associations with the following classes:
Class: DmDdKeyword hasvalidvaluesspecifiedas

4.3.18 ECSCollection Class

Parent Class: Not Applicable
Public: No
Distributed Object: No
Purpose and Description:
This class provides description of the collection.

Attributes:

ArchiveCenter - Center where collection is archived.
Data Type: string(20)
Privilege: Private
Default Value:
Contraints:not null
Non Persisent Flag:False

Description - This is the description of the collection.

ProcessingCenter - Center where collection was or is being processed. i.e. name of DAAC or SCF.
Data Type: string(20)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

RevisionDate - Represents the date and possibly the time that this directory entry was created or the latest date and time of its modification or update.
Data Type: string(20)
Privilege: Private
Default Value:
Contraints:
Non Persisent Flag:False

ShortName - This is a short name for the collection.

StorageMedium -
Data Type: string(30)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

SuggestedUsage - This attribute describes how this collection or granule may be best used to support earth science/global change research.
Data Type: varstring(500)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

Operations:

None

Associations:

The ECSCollection class has associations with the following classes:
Class: ControlledParameter
Class: Discipline
Class: Parameter
Class: PhysicalParameterDetails
Class: Platform
Class: Sensor
Class: Topic
Class: UncontrolledParameter
Class: DmDdInfoMgr providesaccessto

4.3.19 EcCsAsynchRequest_C Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is a Server Frame Work (SRF) object. Refer 305-CD-028-001 for description.

Attributes:

None

Operations:

None

Associations:

The EcCsAsynchRequest_C class has associations with the following classes:
Class: EcCsMsg tracks&distributes

4.3.20 EcCsMsg Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is a Server Frame Work object. Refer 305-Cd-028-001 for description.

Attributes:

None

Operations:

None

Associations:

The EcCsMsg class has associations with the following classes:
Class: DmDdRequestServer_C generates
Class: EcCsAsynchRequest_C tracks&distributes

4.3.21 EcCsRequestServer_C Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is a Server Frame Work object. Refer 305-CD-028-001 for description.

Attributes:

None

Operations:

None

Associations:

The EcCsRequestServer_C class has associations with the following classes:
None

4.3.22 Instrument Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class describes the basic attributes of an instrument which may be either: a) the sole generator of the collection granules or b) an input to field campaigns or non-instrument collections. The instrument is used to collect the data directly, or to gather the data from multiple sensors which comprise it. The operation mode is expected to update frequently compared to the other characteristics. Where an instrument has only one distinguishable sensor (e.g. AVHRR) then sensor and instrument are the same, and shall be named as such.

Attributes:

InstrumentName - The long or full name by which the instrument is commonly known. In V0 Valids matrix this corresponds to the Sensor column.
Data Type: string(80)
Privilege: Private
Default Value:
Contraints:InstrumentName must exist if Instrument class is used.
Non Persisent Flag:False

InstrumentType - The type of instrument.
Data Type: string(20)
Privilege: Private
Default Value:

NumberofSensors - The number of sensors carried by the instrument.
Data Type: int(2)
Privilege: Private
Default Value:
Contraints:NumberofSensors => 1
Non Persisent Flag:False

Operations:

None

Associations:

The Instrument class has associations with the following classes:
Class: Platform
Class: Sensor

4.3.23 InstrumentPlatformXref Class

Parent Class: Not Applicable

Attributes:

InstrrumentType

OperationMode - Mode of operation of the instrument. Each instrument will have 1 to n modes which may be static for the collection, or change on a granule-by-granule basis. (e.g. domains: launch, survival, initialization, safe, diagnostic, roll, tilt, standby, routine, test, calibration).
Data Type: string(20)
Privilege: Private
Default Value:

PlatformShortName - The acronym, abbreviation, or short name assigned to the platform carrying the instrument(s).
Data Type: string(20)
Privilege: Private
Default Value:

Operations:

None

Associations:

The InstrumentPlatformXref class has associations with the following classes:
None

4.3.24 MultipleTypeCollection Class

Parent Class: ECSCollection
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
This class describes collections that are made from multiple other collections. An example of this is an event such as a flood or volcano eruption. The data for these events would be made from possibly more than one SingleTypeCollection.

Attributes:

AggregationAttribute - The attribute on which aggregation occurs.

AggregationType - The type of aggregation, for example, EVENT, PARAMETER, etc.
Contraints:If AggregationValue and AggregationRelationship exist then AggregationType must exist.
Non Persisent Flag:False

AggregationValue - This attribute contains the value associated with the aggregation type. An example may be EVENT (aggregation type) = MIDWEST FLOOD '93 (aggregation value). MIDWEST FLOOD '93 would be the value associated with the event or aggregation type.
Contraints:If AggregationType and AggregationRelationship exist then AggregationValue must exist.
Non Persisent Flag:False

Operations:

All Operations inherited from parent class

Associations:

The MultipleTypeCollection class has associations with the following classes:
Class: SingleTypeCollection references

4.3.25 Parameter Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This class is used to describe all (major) (science) parameters in the collection. This class provides for a detailed level 'data dictionary' description of the data product (i.e. content repeated in each granule). The parameter names should be unique across all collections, reflecting the uniqueness of the parameter(s) in the collection. In addition, product specific attributes can be described (i.e. attributes used to describe the collection additional to those in this model). Also, product specific attributes recorded on a granule by granule basis are described here. The ParameterValue is used in two circumstances: 1) when the value relates to the entire collection 2) at the granule level to record the values of non-core attributes.

Attributes:

ParameterDatatype - Data type of parameter.
Data Type: string(10)
Privilege: Private
Default Value:null
Contraints:If ParameterName exists then ParameterDatatype must exist.
Non Persisent Flag:False

ParameterDescription - This attribute provides a description for the parameter.
Data Type: string(255)
Privilege: Private
Default Value:null
Contraints:If ParameterName exists then ParameterDescription must exist.
Non Persisent Flag:False

ParameterName - This attribute identifies the label which is used to reference characteristics of the object,(collection or granule) which are collection-, granule-, or site-specific, thus are not in the core standard. The implementation of this logical model will use the information populating this class to build collection-specific schemas at each LIM, and the LIM services will use it to decompose and recompose queries on these non-core attributes. i.e. cloud assessment, bands acquired, bands used, elevation, instrument to target distance.
Data Type: string(30)
Privilege: Private
Default Value:null
Contraints:If ParameterDescription exists then ParameterName must exist.
Non Persisent Flag:False

Operations:

None

Associations:

The Parameter class has associations with the following classes:
Class: ECSCollection
Class: PhysicalParameterDetails
Class: UncontrolledParameter

4.3.26 PhysicalParameterDetails Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This class is used to describe parameters in the parameter class when they are physical or geophysical in nature. ~ ~

Attributes:

ParameterMeasurementResolution - This attribute will be used to identify the smallest unit increment to which the parameter value is measured.
Data Type: string(30)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

ParameterRange - This attribute provides maximum and minimum value of parameter over whole collection.
Data Type: string(10)
Privilege: Private
Default Value:
Contraints:
Non Persisent Flag:False

ParameterUnitsofMeasurement - The standard unit of measurement for a non-core attribute. AVHRR: Units of Geophysical Parameter=Units of Geophysical Parameter ~
Data Type: string(20)
Privilege: Private
Default Value:null
Contraints:If ParameterValue exists then ParameterUnitsofMeasurement exist.
Non Persisent Flag:False

ParameterValueAccuracy - An estimate of the accuracy of the assignment of attribute value. i.e. AVHRR: Measurement Error or Precision=Measurement error or precision of a data product parameter. This can be specified in percent or the units with which the parameter is measured.
Data Type: string(30)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

ParameterValueAccuracyExplanation - This defines the method used for determining the Parameter Value Accuracy that is given for this non core attribute.
Data Type: varstring(255)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

Operations:

None

Associations:

The PhysicalParameterDetails class has associations with the following classes:
Class: ECSCollection
Class: Parameter

4.3.27 Platform Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:
Purpose and Description:
Information identifying the carrier for the instruments/sensors providing the measurements.

Attributes:

PlatformLongName - The long name for the platform.
Data Type: string(80)
Privilege: Private
Default Value:null

PlatformShortName - The short name for the platform.
Data Type: string(20)
Privilege: Private
Default Value:null

Operations:

None

Associations:

The Platform class has associations with the following classes:
Class: ECSCollection
Class: Instrument

4.3.28 RWPtrSlist Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is singly-linked list class provided by the RogueWave Tools.h++.

Attributes:

None

Operations:

None

Associations:

The RWPtrSlist class has associations with the following classes:
None

4.3.29 Sensor Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The device actually sensing/measuring the data being collected.

Attributes:

SensorLongName - The long name of the sensor.
Data Type: string(80)
Privilege: Private
Default Value:null

SensorShortName - The short name for the sensor.
Data Type: string(20)
Privilege: Private
Default Value:null

SensorType - The sensor type.
Data Type: string(40)
Privilege: Private
Default Value:null

Operations:

None

Associations:

The Sensor class has associations with the following classes:
Class: ECSCollection
Class: Instrument
Class: SensorCharacteristic

4.3.30 SensorCharacteristic Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This describes individual characteristics of a sensor.

Attributes:

SensorCharacteristicName - The name of the sensor characterstic being described by this object, for example, "channel 1 start spectrum".
Data Type: string(80)
Privilege: Private
Default Value:null

SensorCharacteristicType - The type of characteristic being described in this object.
Data Type: string(20)
Privilege: Private
Default Value:null

SensorCharacteristicUnit - The unit of measurement of the SensorCharacteristicValue associated with a particular sensor characteristic.
Data Type: string(20)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

SensorCharacteristicValue - The value of the sensor characteristic, for example, a channel spectrum value.
Data Type:
Privilege: Private
Default Value:null

Operations:

None

Associations:

The SensorCharacteristic class has associations with the following classes:
Class: Sensor

4.3.31 SingleTypeCollection Class

Parent Class: ECSCollection
Public: No
Distributed Object:No
Purpose and Description:

Attributes:

AccessConstraints - Restrictions and legal prerequisites for accessing the collection. These include any access constraints applied to assure the protection of privacy or intellectual property, and any special restrictions or limitations on obtaining the collection. These restrictions differ from Use Restrictions in that they only apply to access.
Data Type: varstring(255)
Privilege: Private
Default Value:none
Contraints:
Non Persisent Flag:False

CitationforExternalPublication - The recommended reference to be used when referring to this collection in publications. Its format is free text, but should include: Orginator (the name of an organization or individual that developed the data set, where Editor(s)' names are followed by (ed.) and Compiler(s)' names are followed by (comp.)); Publication date (the date of publication or release of the data set); Title (the name by which document can be referenced).
Data Type: varstring(255)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

CollectionState - This attribute describes the state of the collection, whether it is planned but not yet existent, partially complete due to continual additions from remotely sensed data/processing/reprocessing, or is considered a complete product/dataset.
Data Type: varstring(255)
Privilege: Private
Default Value:none
Contraints:
Non Persisent Flag:False

CollectionURL - The URL for the guide document describing the collection (if applicable).

TemporalKeyword - This attribute specifies a word or phrase which serves to summarize the temporal characteristics referenced in the collection. i.e. Monthly Composite, Annual Mean.
Data Type: string(10)
Privilege: Private
Default Value:null
Contraints:
Non Persisent Flag:False

Operations:

All Operations inherited from parent class

Associations:

The SingleTypeCollection class has associations with the following classes:
Class: DmDdOperation hasassociatedservices
Class: DmDdAttribute isdescribedby
Class: MultipleTypeCollection references

4.3.32 Topic Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This class provides the second (of 5) levels of description of the collection content. GCMD keywords are used to describe the general topic area of the collection. A collection can conceivably cover several topics.

Attributes:

TopicKeyword - Keyword used to describe the general topic area of the collection. A collection can conceivably cover several topics.
Data Type: string(32)
Privilege: Private
Default Value:
Contraints:not null
Non Persisent Flag:False

Operations:

None

Associations:

The Topic class has associations with the following classes:
Class: ECSCollection
Discipline (Aggregation)

4.3.33 UncontrolledParameter Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This class provides the fourth (of 5) levels of description of the collection content. GCMD keywords are used to describe the specific science parameter content of the collection. A collection can conceivably cover many specific parameters. New keywords may be added through a managed procedure. The keyword valids are the lowest level physical parameter terms which are normally searchedby a user; i.e. a user enters a keyword which when found may connect with one ormore parameters from collections.

Attributes:

UncontrolledParameterKeyword - Keyword used to describe the specific science parameter content of the collection. A collection can conceivably cover many specific parameters. The keyword valids are the lowest level physical parameter terms which are normally searched by a user; i.e. a user enters a keyword which when found may commect with one or more parameters from collections. The keywords are also the lowest level words which describe product content without being the server specific measurement (held in Parameter class). While there is a controlled list of these parameters held by GCMD, additions can be made by an as yet unspecified configuration control process.
Data Type: string(80)
Privilege: Private
Default Value:
Contraints:not null
Non Persisent Flag:False

Operations:

None

Associations:

The UncontrolledParameter class has associations with the following classes:
Class: ECSCollection
Class: Parameter
ControlledParameter (Aggregation)

4.4 CSCI Structure

Table 4.4-1 provides a summary of the components that make up the DDICT CSCI, to the extent that they are currently known. Since this CSCI is on the incremental development track, the table presents our current best estimate of the CSCI components, but is likely to change as the CSCI evolves.

Table 4.4-1. Data Dictionary CSCI Components

Component Name	Description	Type (DEV or OTS)
Data Dictionary Server	Application server that processes the Data Dictionary requests.	DEV
DDICT Client Library	This is the library that clients use to connect and submit requests to the DDICT server.	DEV
DDICT Request Processing	This is the group of objects that accepts requests from the clients and resolves them. Request types include searches and schema modifications.	DEV
Data Dictionary DBMS	DBMS (Sybase) used to store and provide access to Data Dictionary information.	OTS
Data Dictionary Maintenance Tool	Tool that provides access to maintenance functions of the DDICT service. This tool may need to be separated into two separate CSCs. Refer to section 4.5 for further discussion on this tool.	DEV + OTS

4.5 CSCI Management and Operation

The maintenance and operations functions that are required for the DDICT service fall into two classes:

1. Management of the Data Dictionary Information with respect to the components that are accessed. Consistency checking routines can be run to determine if the information available in the data dictionary is consistent with the schemas that are available at the site. This function would be used in response to repeated failed requests to specific components. For example, if a particular science data server or LIMGR continually returns an error when

queries are posed submitted to it, the operators can execute the consistency checker to determine if the currently available schema at the science data server is consistent with the information stored in the DDICT database. This situation could only occur through some sort of failure scenario. Also manual updates of the data dictionary information may be necessary to ensure the consistency.

2. Maintenance functions on the off-the-shelf DBMS. These functions would be performed for such things as database recovery and backups and other DBMS functions.

It may be necessary to separate these tools based on the role of the operators at the site. If the two functions are performed by distinct user classes it may be better to have two separate tools.

5. DIMGR - Distributed Information Manager CSCI

5.1 CSCI Overview

The DIMGR provides access to data and services accessible from repositories distributed across a wide-area network. The DIMGR decomposes requests and executes the component parts to LIMGRs, Science Data Servers (SDSRV), and/or GTWAYs. The LIMGRs, SDSRVs, and GTWAYs make themselves accessible to the DIMGR by exporting schema information to the DDICT and making services available through the Advertising Service (ADSRV). The access paths possible by a DIMGR is configured by an operator using both the Data Dictionary Maintenance Tool and the DIMGR Configuration Tool. The access paths are also dependent on the configuration of the LIMGRs, SDSRVs, and GTWAYs both at the site of the DIMGR and the other sites in the network. For example, a site may chose to have all access outside the site come through the LIMGR. If this is the case, the operator at another site would not be able to configure his/her DIMGR to access the SDSRV of the other site directly. The Advertising Service controls the access to services.

The current baseline for the configuration of the DIMGR is to have one global DIMGR available at each DAAC. A global DIMGR is one which has access to all data available in the EOSDIS provider network. Sites may define other DIMGRs which provide access to a smaller set of data across the WAN, but this is not required. The driving factors behind providing this type of DIMGR would be to improve performance and provide access to a particular grouping of data of interest to a site’s users.

5.2 CSCI Context

Figure 5.2-1 shows the context diagram for the DIMGR. Table 3.2-1 defines the flows in this diagram.

5.3 DIMGR Object Models

The public classes for the DIMGR are the same as the LIMGR CSCI. The LIMGR CSCI client library is reused in the DIMGR in order to provide a consistent interaction between the Data Managment Subsystem components. Please refer to the LIMGR object models to review the client interface to the DIMGR.

5.4 CSCI Structure

Table 5.3-1 provides a summary of the components that make up the DIMGR CSCI, to the extent that they are currently known. Since this CSCI is on the incremental development track, this is subject to change as the CSCI evolves.

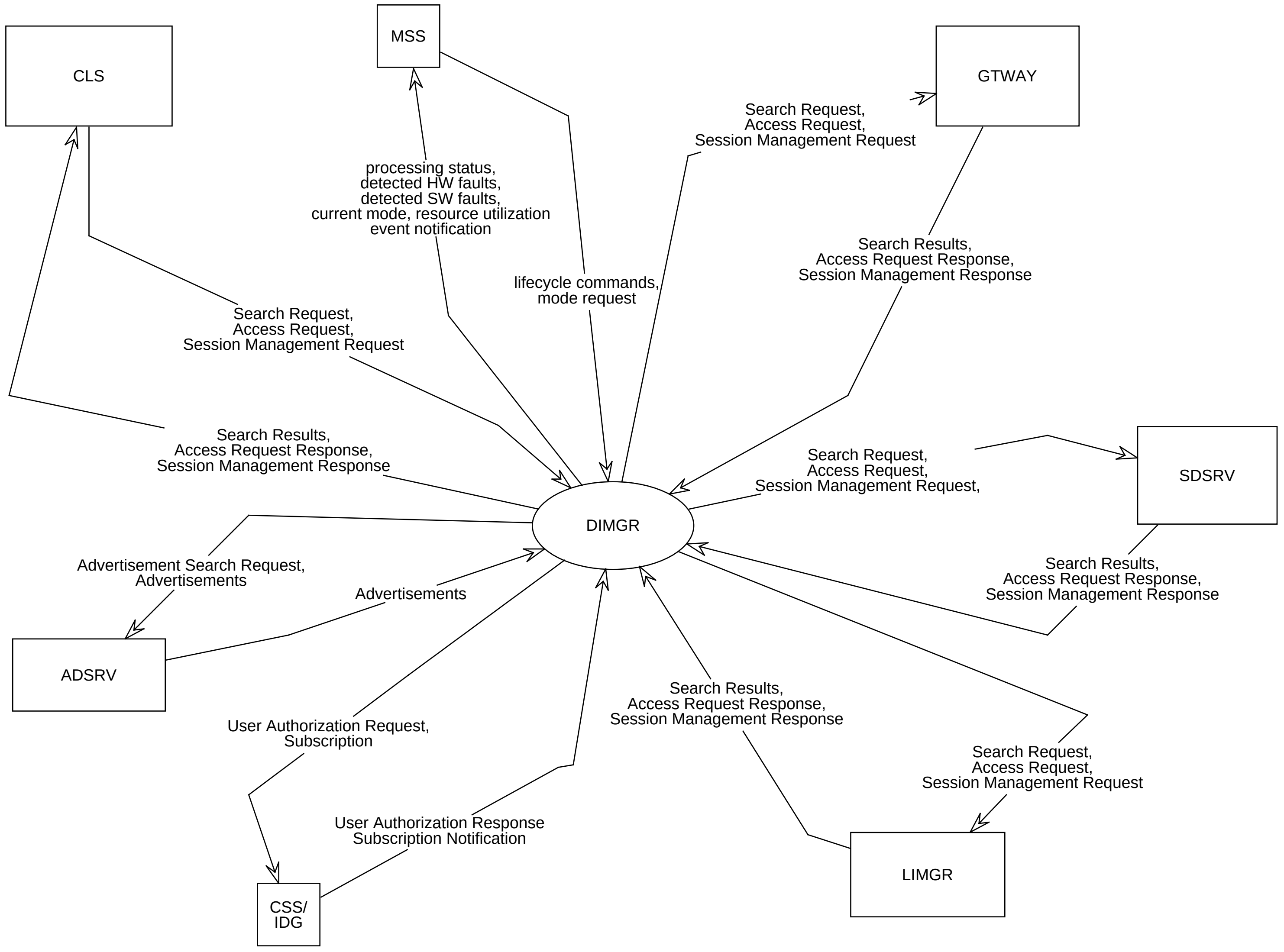

CLS
MSS
GTWAY
SDSRV
ADSRV
LIMGR
CSS/ IDG
DIMGR
processing status, detected HW faults, detected SW faults, current mode, resource utilization event notification
lifecycle commands, mode request
Search Request, Access Request, Session Management Request
Search Results, Access Request Response, Session Management Response
Search Request, Access Request, Session Management Request
Search Results, Access Request Response, Session Management Response
Search Request, Access Request, Session Management Request,
Search Results, Access Request Response, Session Management Response
Advertisement Search Request, Advertisements
Advertisements
User Authorization Request, Subscription
User Authorization Response Subscription Notification
Search Results, Access Request Response, Session Management Response
Search Request, Access Request, Session Management Request

Figure 5.2-1. Distributed Information Manager Context

Table 5.3-1. DIMGR CSCI Components

Component Name	Description	Type (DEV or OTS)
DIMGR Server	Application server that processes the DIMGR requests.	DEV
Configuration/Setup	This is the GUI interface to DIMGR configuration.	DEV

5.5 CSCI Management and Operation

The Data Dictionary Maintenance Tool is used by operators to configure the scope of the DIMGR's data and service access. When the schema for a new data collection is defined at a SDSRV, it is exported to the DDICT service. The DIMGR can be configured to automatically incorporate this new data collection in the DIMGR's scope. This would be accomplished through a subscription on the DDICT information. If the DIMGR were configured to automatically accept any new data collection, it would submit a subscription for DDICT information upon startup. It would then listen for notifications of updates to the DDICT. When a subscription notification arrived, the DIMGR would reread the DDICT service data and update the InfoMgr object to create a relationship to the new data collection for the DIMGR.

Another alternative under operator control is to have a manual incorporation of new data collections in the DIMGR. With this method, the operator would receive notification of a new data collection through e-mail. He/she would then use the Data Dictionary Maintenance Tool to review the new data collection. If the operator wanted to accept the new data collection as part of the DIMGR, he/she would update the InfoMgr object to create the relationship between the DIMGR and the new data collection.

The DIMGR Configuration and Setup Tool is used to set the options for the DIMGR. In addition to the option for automatic or manual incorporation of new collections, the DIMGR Configuration and Setup Tool can be used to manually force a reread of the DDICT service. This would be used in the manual incorporation mode, when the operator wanted the changes to the DDICT service to take effect in the DIMGR. Another function of the DIMGR Configuration and Setup Tool is to define run-time parameters such as maximum number of sessions, result set memory cache per session, etc.

This page intentionally left blank.

6 LIMGR - Local Information Manager CSCI

6.1 CSCI Overview

The goal of the LIMGR is to provide access to a site while hiding the underlying details of the site architecture. The LIMGR accepts search and data access requests and issues these to other search agents (such as Science Data Servers). Science Data Servers make themselves available to the LIMGR by exporting their schema information to the DDICT. An operator adds the Science Data Server to the LIMGR's federation using the Data Dictionary Tool described in Section 4.5 of this document. The LIMGR makes itself available to DIMGRs by exporting its schema to the DDICT service. It also advertises its services to the Advertising Service.

The LIMGR can provide the sole interface to a site or the operators can make the Science Data Servers directly available as well. The number of LIMGRs is site dependent at this point, but the baseline is that there will be one LIMGR per DAAC. This LIM will provide access to all the Science Data Servers at that DAAC as well as access to the V0 Gateway.

6.2 Local Information Manager Context

Figure 6.2-1 shows the context diagram for the LIMGR CSCI. Table 3.2-1 defines the flows on this diagram.

6.3 Local Information Manager Object Model

The public interface of the LIMGR is reused in the DIMGR and GTWAY CIs. The object model in Figure 6.3-1 shows the client library used to interface to the LIMGR, DIMGR, and GTWAY. The LIMGR client library is based on the Server Request Framework (SRF). It provides three objects that are subclassed off the SRF that the client programs will have to interact with. The DmImClRequestServer is a the object which maintains a synchronous connection with the server. It is used by the client to create new asynchronous request objects, DmImClRequest. The DmImClAdmRequestServer is the administration request server object. This object contains mostly operations that require access privileges in order to execute them. The message objects, such as DmImBrowseMsg are classes that the client does not directly interact with. The request server and asynchronous request objects use them to pass information back and forth across the network.

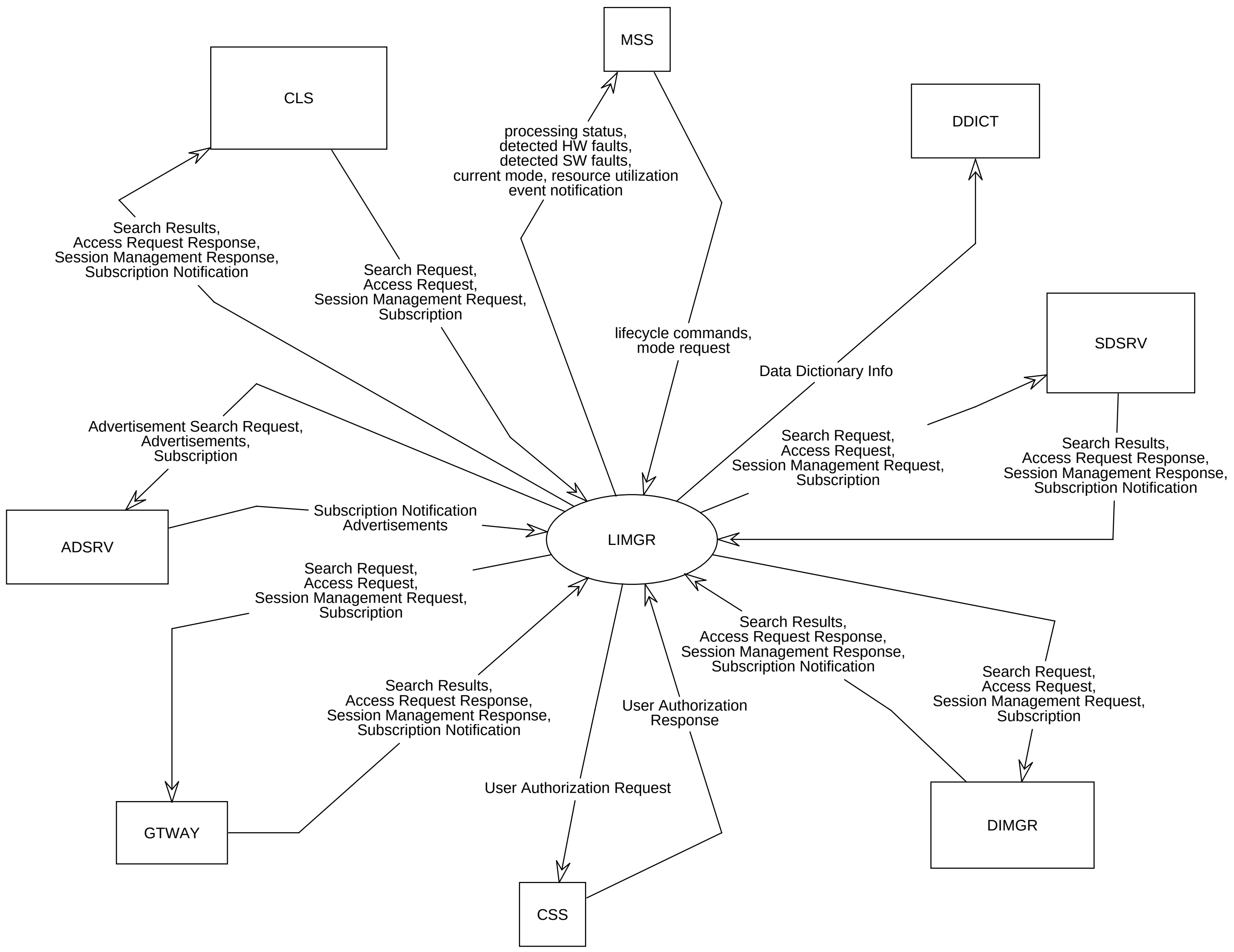
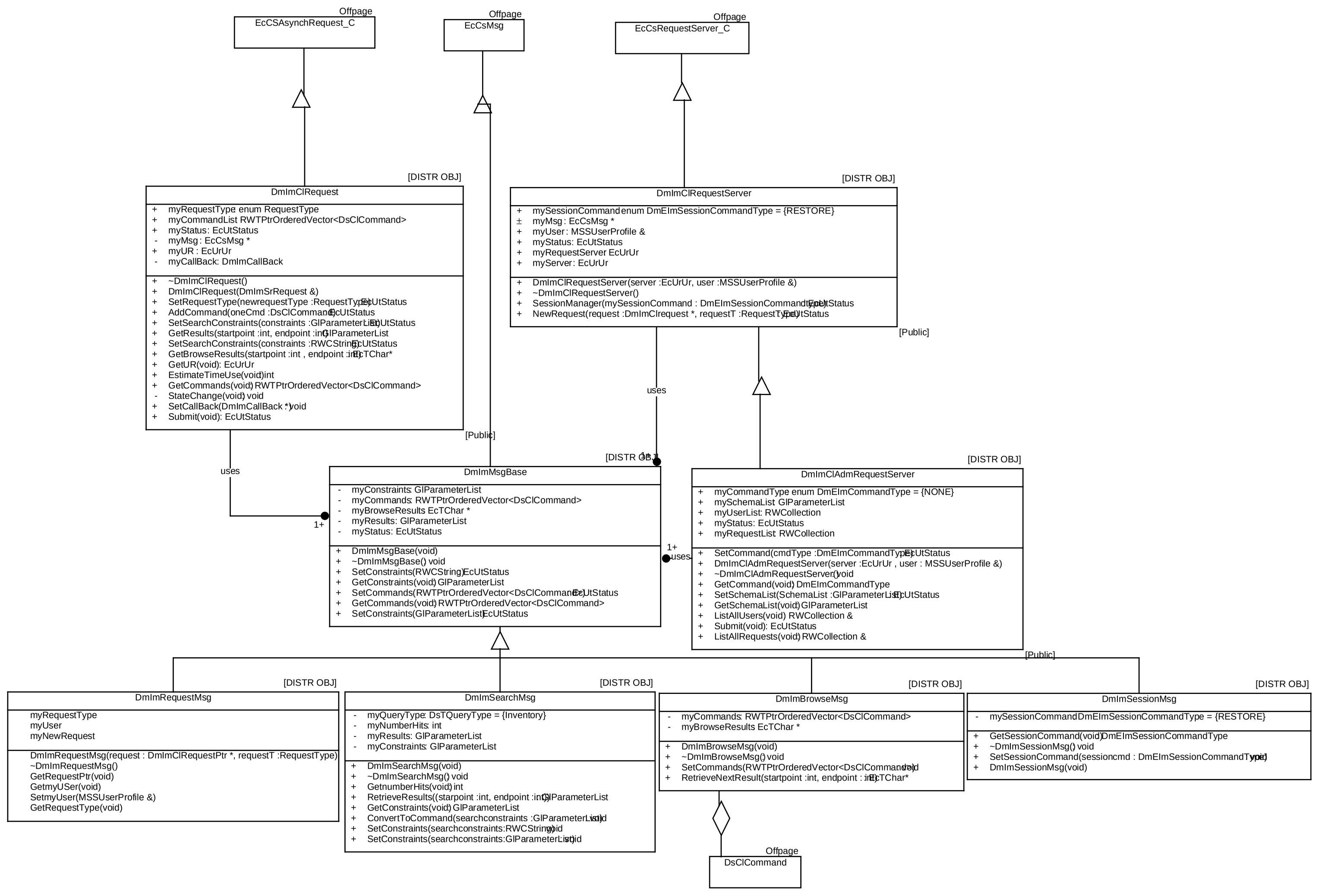


Figure 6.2-1 LIMGR CSCI Context Diagram



Offpage
EcCSAsynchRequest_C
Offpage
EcCsMsg
Offpage
EcCsRequestServer_C
[DISTR OBJ]
DmImClRequest
+ myRequestType enum RequestType
+ myCommandList RWTPtrOrderedVector<DsClCommand>
+ myStatus: EcUtStatus
- myMsg : EcCsMsg *
+ myUR : EcUrUr
- myCallBack: DmImCallBack
+ ~DmImClRequest()
+ DmImClRequest(DmImSrRequest &)
+ SetRequestType(newrequestType :RequestType) EcUtStatus
+ AddCommand(oneCmd :DsClCommand) EcUtStatus
+ SetSearchConstraints(constraints :GlParameterList) EcUtStatus
+ GetResults(startpoint :int, endpoint :int) GlParameterList
+ SetSearchConstraints(constraints :RWCString) EcUtStatus
+ GetBrowseResults(startpoint :int , endpoint :int) EcTChar*
+ GetUR(void): EcUrUr
+ EstimateTimeUse(void) int
+ GetCommands(void) RWTPtrOrderedVector<DsClCommand>
- StateChange(void) void
+ SetCallBack(DmImCallBack *) void
+ Submit(void): EcUtStatus
[Public]
uses
1+
[DISTR OBJ]
DmImClRequestServer
+ mySessionCommand enum DmEImSessionCommandType = {RESTORE}
± myMsg : EcCsMsg *
+ myUser : MSSUserProfile &
+ myStatus: EcUtStatus
+ myRequestServer EcUrUr
+ myServer: EcUrUr
+ DmImClRequestServer(server :EcUrUr, user :MSSUserProfile &)
+ ~DmImClRequestServer()
+ SessionManager(mySessionCommand : DmEImSessionCommandType) EcUtStatus
+ NewRequest(request :DmImClrequest *, requestT :RequestType) EcUtStatus
[Public]
uses
1+
[DISTR OBJ]
DmImMsgBase
- myConstraints: GlParameterList
- myCommands: RWTPtrOrderedVector<DsClCommand>
- myBrowseResults EcTChar *
- myResults: GlParameterList
- myStatus: EcUtStatus
+ DmImMsgBase(void)
+ ~DmImMsgBase() void
+ SetConstraints(RWCString) EcUtStatus
+ GetConstraints(void) GlParameterList
+ SetCommands(RWTPtrOrderedVector<DsClCommand>) EcUtStatus
+ GetCommands(void) RWTPtrOrderedVector<DsClCommand>
+ SetConstraints(GlParameterList) EcUtStatus
1+
uses
[DISTR OBJ]
DmImClAdmRequestServer
+ myCommandType enum DmEImCommandType = {NONE}
+ mySchemaList GlParameterList
+ myUserList: RWCollection
+ myStatus: EcUtStatus
+ myRequestList: RWCollection
+ SetCommand(cmdType :DmEImCommandType) EcUtStatus
+ DmImClAdmRequestServer(server :EcUrUr , user : MSSUserProfile &)
+ ~DmImClAdmRequestServer() void
+ GetCommand(void) DmEImCommandType
+ SetSchemaList(SchemaList :GlParameterList) EcUtStatus
+ GetSchemaList(void) GlParameterList
+ ListAllUsers(void) RWCollection &
+ Submit(void): EcUtStatus
+ ListAllRequests(void) RWCollection &
[Public]
[DISTR OBJ]
DmImRequestMsg
myRequestType
myUser
myNewRequest
DmImRequestMsg(request : DmImClRequestPtr *, requestT :RequestType)
~DmImRequestMsg()
GetRequestPtr(void)
GetmyUSer(void)
SetmyUser(MSSUserProfile &)
GetRequestType(void)
[DISTR OBJ]
DmImSearchMsg
- myQueryType: DsTQueryType = {Inventory}
- myNumberHits: int
- myResults: GlParameterList
- myConstraints: GlParameterList
+ DmImSearchMsg(void)
+ ~DmImSearchMsg() void
+ GetnumberHits(void) int
+ RetrieveResults((starpoint :int, endpoint :int) GlParameterList
+ GetConstraints(void) GlParameterList
+ ConvertToCommand(searchconstraints :GlParameterList) void
+ SetConstraints(searchconstraints:RWCString) void
+ SetConstraints(searchconstraints:GlParameterList) void
[DISTR OBJ]
DmImBrowseMsg
- myCommands: RWTPtrOrderedVector<DsClCommand>
- myBrowseResults EcTChar *
+ DmImBrowseMsg(void)
+ ~DmImBrowseMsg() void
+ SetCommands(RWTPtrOrderedVector<DsClCommand>) void
+ RetrieveNextResult(startpoint :int, endpoint : int) EcTChar*
Offpage
DsClCommand
[DISTR OBJ]
DmImSessionMsg
- mySessionCommand DmEImSessionCommandType = {RESTORE}
+ GetSessionCommand(void) DmEImSessionCommandType
+ ~DmImSessionMsg() void
+ SetSessionCommand(sessioncmd : DmEImSessionCommandType) void
+ DmImSessionMsg(void)

6.3-1. DmImClient Object Model Diagram

6.3.1 DmImBrowseMsg Class

Parent Class: DmImMsgBase
Public: No
Distributed Object:Yes
Purpose and Description:
This message class is specialized from DmImMsgBase. It encapsulate the methods and datatypes specific to handle Browse requests and Browse results.

Attributes:

myBrowseResults - Contains the result set specified by the caller .
Data Type: EcTChar *
Privilege: Private
Default Value:

myCommands - Contains the list of DsClCommand obtained from the request object. This list once shipped with the message object will be accessible by the server side request.
Data Type: RWTPtrOrderedVector<DsClCommand>
Privilege: Private
Default Value:

Operations:

DmImBrowseMsg - Default constructor for DmImBrowseMsg
Arguments: void
Return Type: Void
Privilege: Public

RetrieveNextResult - This method retrieves the result specified by startpoint and enpoint and returns it to the caller.
Arguments: startpoint :int, endpoint : int
Return Type: EcTChar*
Privilege: Public

SetCommands - The request object calls this method when it wants to have the entire set of commands packaged in the message object. The commands will then be ready to be shipped to the server side.
Arguments: RWTPtrOrderedVector<DsClCommand>
Return Type: void
Privilege: Public

~DmImBrowseMsg - Default destructor for DmImBrowseMsg
Arguments:
Return Type: void
Privilege: Public

Associations:

The DmImBrowseMsg class has associations with the following classes:
Class: DsClCommand

6.3.2 DmImClAdmRequestServer Class

Parent Class: DmImClRequestServer
Public: Yes
Distributed Object:Yes
Purpose and Description:
This object is used for sending administration command to the server side and receiving information from the server. It is synchronously connected to the server .

Attributes:

myCommandType - Enumerated type which contains the command to be applied to the schemaList. The enumeration is {UPDATE, DELETE, ADD }
Data Type: enum DmEImCommandType
Privilege: Public
Default Value:{NONE}

myRequestList - ROGUE WAVE Collection that contains all the requests that are being tracked by the server.
Data Type: RWCollection
Privilege: Public
Default Value:

mySchemaList - GlParameterList that contains the information to modify the schema of the LIM. It will be acted upon by myCommandType at the server side.
Data Type: GlParameterList
Privilege: Public
Default Value:

myStatus - status flag returned after SetCommand and SetSchemaList .This enables the caller to check whether the functions performed their actions without errors.

Data Type:	EcUtStatus
Privilege:	Public
Default Value:	

myUserList - Rogue Wave Collection of all the users that have a request pending at the server side.

Data Type:	RWCollection
Privilege:	Public
Default Value:	

Operations:

DmImClAdmRequestServer

Arguments:	server :EcUrUr , user : MSSUserProfile &
Return Type:	Void
Privilege:	Public

GetCommand - This method will return the command type to the caller

Arguments:	void
Return Type:	DmEImCommandType
Privilege:	Public

GetSchemaList - This method will return the SchemaList to the caller as a GlParameterList

Arguments:	void
Return Type:	GlParameterList
Privilege:	Public

ListAllRequests - This method will return to the caller a reference to the collection of all the requests that are being tracked at the server.

Arguments:	void
Return Type:	RWCollection &
Privilege:	Public

ListAllUsers - This method will return to the caller a reference to the collection of all the users that have a pending request at the server.

Arguments:	void
Return Type:	RWCollection &
Privilege:	Public

SetCommand
Arguments: cmdType :DmEImCommandType
Return Type: EcUtStatus
Privilege: Public

SetSchemaList - this method will set the SchemaList provided by the caller into a local Glparameter List
Arguments: SchemaList :GlParameterList
Return Type: EcUtStatus
Privilege: Public

Submit - This method will check to see if CommandType are set and SchemaList is set then will create a new message object package the paramters and send that message to the server for processing. The message passing and processing is handle accoeding to the Server Request Framework Paragdim.
Arguments: void
Return Type: EcUtStatus
Privilege: Public

```
PDL:If myCommandType != NONE
  If mySchemaList != 0
    myMsg *= new DmImMsgBase(myCommandType, mySchemaList)
    myHandler.SendRAcceptance (myMSg)
    // myHandler is inherited from EcCsRequestServer
    // myHandler is the EcCsMsgHandler of the Server Reuest
    // Framework.
  endif
endif
```

~DmImClAdmRequestServer - Basic destructor. when invoked all message objects and the connection to the server will be destroyed.
Arguments:
Return Type: void
Privilege: Public

Associations:

The DmImClAdmRequestServer class has associations with the following classes:
Class: DmImMsgBase uses

6.3.3 DmImClRequest Class

Parent Class: EcCSAsynchRequest_C
Public: Yes
Distributed Object:Yes
Purpose and Description:
This class will handle all requests submitted by the caller . It is part of the Server Request Framework by inheriting from EcCSAynchRequest_C.

Attributes:

myCallBack - This attribute is a pointer of type DmImCallback to a function provided by the caller.
Data Type: DmImCallBack
Privilege: Private
Default Value:

myCommandList - Contains the collection of commands created by the caller.
Data Type: RWTPtrOrderedVector<DsClCommand>
Privilege: Public
Default Value:

myMsg - Points to the message class that is used by the request to communicate and pass parameters to the server side.
Data Type: EcCsMsg *
Privilege: Private
Default Value:

myRequestType - Caller specified which request type he is using. That information will be needed to properly process the parameters that are associated with the request
Data Type: enum RequestType
Privilege: Public
Default Value:

myStatus - Contains status information returned to the caller when calling a method
Data Type: EcUtStatus
Privilege: Public
Default Value:

myUR - Contains the Unique reference of the request. This information needs to be be available to the caller to allow the caller to differentiate a request from another one.
Data Type: EcUrUr
Privilege: Public
Default Value:

Operations:

AddCommand - Will allow the caller to add a command to the the list of commands contained in myCommandList. The entire set is passed to the message class when submit is called.

Arguments:	oneCmd :DsClCommand
Return Type:	EcUtStatus
Privilege:	Public

DmImClRequest - The construction of DmImClRequest is made at the server side at the time DmImSrRequest is being created. All the initialization informations are extracted from the server side object. That is why the argument for the constructor of DmImClRequest is a reference to the server side request DmImSrRequest.

Arguments:	DmImSrRequest &
Return Type:	Void
Privilege:	Public

EstimateTimeUse - This method will send a special query to the server side to request an estimate on the time expected to be used to complete the query. The information will be packaged with the query and the return will be made through the same message object. The number returned correspond to the number of seconds.

Arguments:	void
Return Type:	int
Privilege:	Public

GetBrowseResults - This method will allow the caller to retrieve search results . Specified are the startpoint and the endpoint which allow for a range of results to be returned. This allows to customize the size of the results to the caller's workstation capabilities. The type of result returned is EctChar *.

Arguments:	startpoint :int , endpoint :int
Return Type:	EcTChar*
Privilege:	Public

GetCommands - This method will be used when the caller needs to retrieve from the query a set of commands following the DsClCommandBase paradigm.

Arguments:	void
Return Type:	RWTPtrOrderedVector<DsClCommand>
Privilege:	Public

GetResults - This method will allow the caller to retrieve search results . Specified are the startpoint and the endpoint which allow for a range of results to be returned. This allows callers to customize the size of the results to the caller's workstation capabilities.
Arguments: startpoint :int, endpoint :int
Return Type: GlParameterList
Privilege: Public

GetUR - This method will allow the caller to get a DmImClrequest UR returned. It will be used to identify one request from another.
Arguments: void
Return Type: EcUrUr
Privilege: Public

SetCallBack - Allows the caller that created the request to be notified when a state change happens within the request. A state change for example will occur when the request has completed its query and the results are available. The function supplied has to accept two parameters, myUR : EcUrUr and myState, an enumerated type inherited from SRF. myUR will be used by the caller to identify which request is calling back.
Arguments: DmImCallBack *
Return Type: void
Privilege: Public
PDL:SetCallback (funcptr)
myCallback = funcptr
// funcptr is of type DmImCallBack :pointer to a function.

SetRequestType - Will allow the caller to change the request type. This is usefull when the same request is being updated to provide additional services . As an example a BROWSE request would be updated to become an ACQUIRE request.
Arguments: newrequestType :RequestType
Return Type: EcUtStatus
Privilege: Public

SetSearchConstraints - This method will accept a GlParameterlist as argument and pass this argument to the message class . It returns a status of the operation. The Message class is used within the SRF for asynchronous communication between the caller and the Server.
Arguments: constraints :GlParameterList
Return Type: EcUtStatus
Privilege: Public

SetSearchConstraints - This method will accept RWCString as search constraint and pass the argument to the message class. It returns a status of the operation.
Arguments: constraints :RWCString
Return Type: EcUtStatus
Privilege: Public

StateChange - This method will set the correct state for the request object and will invoke the function supplied by the caller and referred by the pointer DmImCallback and pass two parameters. myUR :EcUrUr which identifies the request and the state : an enumerated type inherited from SRF that is used to reveal the state of the request

Arguments:	void
Return Type:	void
Privilege:	Private

```
PDL:mystate = newstate;
// inherited from SRF and newstate obtained from SRF message class
*funcptr(myUR,myState);
//invoke the function provide by SetCallback and pass the UR id and the state of that request.
```

Submit - When caller invokes submit, the request will finish to encapsulate all commands or constraints into a message object and ship that message object to the server side where it will be processed. The message object is determined by the request type and it is shipped using the EcCsMsgHandler object (SRF).

Arguments:	void
Return Type:	EcUtStatus
Privilege:	Public

```
PDL:Switch (myRequestType)
case BROWSE
            myMsg*= new DmImBrowseMsg()
            myMsg->SetCommand(myCommandList)
        myHandler.SendRAcceptance(myMsg)
case SEARCH
            myMsg*= new DmImSearchMsg()
            myMsg->SetConstraints(mySearchConstraints)
            myHandler.SendRAcceptance(myMsg)
case ACQUIRE
            myMsg*= new DmImSearchMsg()
            myMsg->SetCommand(myCommandList)
            myHandler.SendRAcceptance(myMsg)
```

~DmImClRequest - When called the request and the message associated with it will be destroyed.

Arguments:	
Return Type:	Void
Privilege:	Public

Associations:

The DmImClRequest class has associations with the following classes:
Class: DmImMsgBase uses

6.3.4 DmImClRequestServer Class

Parent Class: EcCsRequestServer_C
Public: Yes
Distributed Object:Yes
Purpose and Description:
This object is used for creating requests and to send session command to the server side. It is synchronously connected to the server and its UR is used as a session Id.

Attributes:

myMsg - Pointer to the message class that is used by DmImClRequestServer to communicate and pass parameters to the server side.
Data Type: EcCsMsg *
Privilege: Protected
Default Value:

myRequestServer - Contains the UR that identified the ServerRequest Object. Since there is one Server request object per client per server connection the ID of that object is used as a session ID between the caller and the server.
Data Type: EcUrUr
Privilege: Public
Default Value:

myServer - Contains the UR that enables the serverRequest object to establish the connection with the server.
Data Type: EcUrUr
Privilege: Public
Default Value:

mySessionCommand - Enumerated type that contains the basic command that applies to session management. The default value is a RESTORE which is applied when the caller first connects to the server.
Data Type: enum DmEImSessionCommandType
Privilege: Public
Default Value:{RESTORE}

myStatus - Returned to the caller after a function call. Allows the caller to verify that function operated properly
Data Type: EcUtStatus
Privilege: Public
Default Value:

myUser - This identifies for which user the DmImClrequestServer is created. That information will be used at the server side to check for permissions and to tag the request for each user.
Data Type: MSSUserProfile &
Privilege: Public
Default Value:

Operations:

DmImClRequestServer - This server class is inheriting from EcCsRequestServer . It is synchronously connected to the DmImSrRequestServer which server ID was provided at construction. It is used to manage the user session and to create objects capable of asynchronous communication between the client and server.
Arguments: server :EcUrUr, user :MSSUserProfile &
Return Type: Void
Privilege: Public
PDL:Connect (server, user)
// inherited from SRF
myMsg *= new DmImSessionMsg(user)
// applies default RESTORE command
myHandler.SendMsgRAcceptance(myMsg)

NewRequest - The caller initiates the creation of a newrequest using the DmImClRequest Server. It needs to pass a pointer to that object. That pointer will be assign to Null until the request is build. The pointer will then be assigned to that request and is ready for use. myRequestType will be used to differentiate Queries from Browse from Acquire. Eventually it may be possible by the use of typed constructors to have this step transparent to the caller but for the time being it is expected that different type of requests may use the same constructor but may need different overloaded methods later on. Therefore the type of request needs to be specified. NewRequest will build a message object DmImRequestMsg , encapsulate information about the request to be, and use the EcCsMsgHandler to ship that message to the DmImSrRequestServer. Note that this is a necessary step in creating an asynchronous request. The message object is nothing more than a courier which contains the information allowing both side to coordinate their efforts.
Arguments: request :DmImClrequest *, requestT :RequestType
Return Type: EcUtStatus

Privilege: Public
PDL: myMsg* = DmImRequestMsg(request,RequestT)
myHandler.SendRAcceptance(myMsg)
// myHandler is inherited from EcCsRequestServer from SRF
return(status)

SessionManager
Arguments: mySessionCommand : DmEImSessionCommandtype
Return Type: EcUtStatus
Privilege: Public

~DmImClRequestServer - Disconnect from the server , All DmImClRequest Objects are destroyed. All Message objects available at the client side are destroyed.
Arguments:
Return Type: Void
Privilege: Public
PDL: No PDL

Associations:

The DmImClRequestServer class has associations with the following classes:
Class: DmImMsgBase uses

6.3.5 DmImMsgBase Class

Parent Class: EcCsMsg
Public: No
Distributed Object:Yes
Persistent Class:False
Purpose and Description:

Attributes:

myBrowseResults - contains the result set to be retrieved by the caller for the case of a browse request.
Data Type: EcTChar *
Privilege: Private
Default Value:

myCommands - contains a list of command to be passed to the request on the server side for further processing
Data Type: RWTPtrOrderedVector<DsClCommand>
Privilege: Private
Default Value:

myConstraints - Contains Search constraints passed either as a GlparameterList or a RWCString
Data Type: GlParameterList
Privilege: Private
Default Value:

myResults - Contains the result set to be retrieved by the caller for the case of a search request.
Data Type: GlParameterList
Privilege: Private
Default Value:

myStatus - Returned to the caller after a function call. Allows the caller to verify that function operated properly
Data Type: EcUtStatus
Privilege: Private
Default Value:

Operations:

DmImMsgBase
Arguments: void
Return Type: Void
Privilege: Public

GetCommands - This method will return the entire list of available commands to the caller
Arguments: void
Return Type: RWTPtrOrderedVector<DsClCommand>
Privilege: Public

GetConstraints - This method will return to the caller the search constraints as a GlparameterList
Arguments: void
Return Type: GlParameterList
Privilege: Public

SetCommands - This method will accept a list of command from the caller.
Arguments: RWTPtrOrderedVector<DsClCommand>
Return Type: EcUtStatus
Privilege: Public

SetConstraints - This method will encapsulate the search constraints in the message object. The results will be shipped back via the EcCsMsgHandler . The constraints are received as a RWCString , so as to support a simple ESQL query.
Arguments: RWCString
Return Type: EcUtStatus
Privilege: Public

SetConstraints - This method will encapsulate the search constraints in the message object. The results will be shipped back via the EcCsMsgHandler . The constraints are received as a GlParameterList, so as to support existing implementations of search constraints.
Arguments: GlParameterList
Return Type: EcUtStatus
Privilege: Public

~DmImMsgBase - Default destructor for DmImMsgBase Object
Arguments:
Return Type: void
Privilege: Public

Associations:

The DmImMsgBase class has associations with the following classes:
Class: DmImClAdmRequestServer uses
Class: DmImClRequest uses
Class: DmImClRequestServer uses

6.3.6 DmImRequestMsg Class

Parent Class: DmImMsgBase
Public: No
Distributed Object:Yes
Purpose and Description:
this message class will handle the basic communications between the client and the server. It is specialized to create a new request and send the new request information to the server.

Attributes:

myNewRequest - This contains the address from which the caller will be able to access the new request. The Server will use this address and assign the newly created request object to it.

myRequestType - This information contains the type of request the caller wants. It is set when the request object is constructed.

myUser - This is the user information. this is initialized by the DmImClRequestServer after the request message object has been constructed.

Operations:

DmImRequestMsg - Constructor of the request message . It needs to take 2 arguments , the address to the new request (passed as a pointer to DmImClRequest) and the request type (Searh or Browse or Acquire). Depending of the request type different message object will be used by the request. Note : message objects are key components of the Server Request Framework.
Arguments: request : DmImClRequestPtr *, requestT :RequestType

GetRequestPtr - Allows the caller to get the DmImClRequest pointer returned. This is used to identified which request the caller is using.
Arguments: void

GetRequestType - Allows the return of the request type when called
Arguments: void

GetmyUSer - Allows the caller to get the MSSUserProfile reference returned.
Arguments: void

SetmyUser - Allows for DmImClRequest server to pass the user information to the request message object.
Arguments: MSSUserProfile &

~DmImRequestMsg - This message request is detroyed when destructor is invoked all contained data is lost.
Arguments:

Associations:

The DmImRequestMsg class has associations with the following classes:
None

6.3.7 DmImSearchMsg Class

Parent Class: DmImMsgBase
Public: No
Distributed Object:Yes
Purpose and Description:
This class is specialized from DmImMsgBase. It allows for the manipulation of search queries and provides the method to handle the data type associated with a search query.

Attributes:

myConstraints - This contains the search constraints received by the request object . It is stored as a GlParameterList and will be given to the server as a GlparameterList.
Data Type: GlParameterList
Privilege: Private
Default Value:

myNumberHits - Returns the number of items found by the search. This information can be used by the caller to determine if it can handle possible large amount of data.
Data Type: int
Privilege: Private
Default Value:

myQueryType - This attribute is taken from the Data Server subsystem and allows to specify which type of search the query will do. By Default the query is an inventory search.

Data Type:	DsTQueryType
Privilege:	Private
Default Value:	{Inventory}

myResults - Contains the results as a GlparameterList . The size of results is determined by the startpoint and enpoint parameters provided by the caller when calling Getresults from the request object.

Data Type:	GlParameterList
Privilege:	Private
Default Value:	

Operations:

ConvertToCommand - This method is taken from DsClRequest and allows to convert GlParamerList search constraints in a command format directly usable by the target data server.

Arguments:	searchconstraints :GlParameterList
Return Type:	void
Privilege:	Public

DmImSearchMsg - Default constructor for DmImSearchMsg

Arguments:	void
Return Type:	Void
Privilege:	Public

GetConstraints - This method returns the search constraints as a GlParameterList. it is intended to be mainly used at the server side but can also be used by the request object to update the search constraints.

Arguments:	void
Return Type:	GlParameterList
Privilege:	Public

GetnumberHits - This method returns the number of hits the search request found. This allows the caller to size the amount it can handle at a time. For a large result set the caller may have to call Getresults several time to sample all the results.

Arguments:	void
Return Type:	int
Privilege:	Public

RetrieveResults - This method is called by the request object to obtain the result set specified by the caller with startpoint and endpoint. The message is sent to the server side where it will collect the result sample and bring it back to the request.

Arguments:	(starpoint :int, endpoint :int)
Return Type:	GlParameterList
Privilege:	Public

SetConstraints - This method is called by the request object and is passed a RWCString as argument to populate myConstraint. It will be packaged ina GlparameterList.

Arguments:	searchconstraints:RWCString
Return Type:	void
Privilege:	Public

SetConstraints - This method is called by the request object and will set to myConstraints the search constraints received as arguments.

Arguments:	searchconstraints:GlParameterList
Return Type:	void
Privilege:	Public

~DmImSearchMsg - Default destructor for DmImSearchMsg object.

Arguments:	
Return Type:	void
Privilege:	Public

Associations:

The DmImSearchMsg class has associations with the following classes:
None

6.3.8 DmImSessionMsg Class

Parent Class:	DmImMsgBase
Public:	No
Distributed Object:	Yes

Purpose and Description:

Attributes:

mySessionCommand - Contains the session command assigned by caller. By default the command is RESTORE.

Data Type:	DmEImSessionCommandType
Privilege:	Private
Default Value:	{RESTORE}

Operations:

DmImSessionMsg - Default constructor for the DmImSessionMsg.

Arguments:	void
Return Type:	Void
Privilege:	Public

GetSessionCommand - This method will return the session command to the caller.

Arguments:	void
Return Type:	DmEImSessionCommandType
Privilege:	Public

SetSessionCommand - Allows the caller to set a session command of the type DmEImSessionCommand.

Arguments:	sessioncmd : DmEImSessionCommandType
Return Type:	void
Privilege:	Public

~DmImSessionMsg - Default Destructor of the session message object.

Arguments:	
Return Type:	void
Privilege:	Public

Associations:

The DmImSessionMsg class has associations with the following classes:

None

6.3.9 DsClCommand Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:

Attributes:

None

Operations:

None

Associations:

The DsClCommand class has associations with the following classes:
Class: DmImBrowseMsg

6.3.10 EcCSAsynchRequest_C Class

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The EcCSAsynchRequest_C class has associations with the following classes:
None

6.3.11 EcCsMsg Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is a Server Frame Work object. Refer 305-Cd-028-001 for description.

Attributes:

None

Operations:

None

Associations:

The EcCsMsg class has associations with the following classes:
None

6.3.12 EcCsRequestServer_C Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
This is a Server Frame Work object. Refer 305-CD-028-001 for description.

Attributes:

None

Operations:

None

Associations:

The EcCsRequestServer_C class has associations with the following classes:
None

6.4 CSCI Structure

Table 6.4-1 provides a summary of the components that make up the LIMGR CSCI, to the extent that they are currently known. Since this CSCI is on the incremental development track, this is subject to change as the CSCI evolves.

Table 6.4-1. LIMGR CSCI Components

Component Name	Description	Type (DEV or OTS)
LIMGR Server	Application server that processes the LIMGR requests.	DEV
Client Library	The Client library used by programs to connect and submit requests to the LIMGR.	DEV
Request Processing	The CSC that takes a request and builds a plan for executing the request.	DEV
Mapping Layer	The CSC that maps from ECS terms and attributes to non-ECS terms and attributes	DEV
Database Interface	This is wrapper class around the COTS DBMS. It provides for insertion of other DBMSs with minimal code impact.	DEV
External Interface	This is the interface from the LIMGR to other search agents, such as the Data Server, and GTWAY.	DEV
Configuration and Setup Tool	This is a GUI that helps the operator configure the LIMGR>	DEV

6.5 CSCI Management and Operation

The Data Dictionary Maintenance Tool is used by operators to configure the scope of the LIMGR's data and service access. When the schema for a new data collection is defined at a SDSRV, it is exported to the DDICT service. The LIMGR can be configured to automatically incorporate this new data collection in the LIMGR's scope. This would be accomplished through a subscription on the DDICT information. If the LIMGR were configured to automatically accept any new data collection, it would submit a subscription for DDICT information upon startup. It would then listen for notifications of updates to the DDICT. When a subscription notification arrived, the LIMGR would reread the DDICT service data and update the InfoMgr object to create a relationship to the new data collection for the LIMGR.

Another alternative under operator control is to have a manual incorporation of new data collections in the LIMGR. With this method, the operator would receive notification of a new data collection through e-mail. He/she would then use the Data Dictionary Maintenance Tool to review the new data collection. If the operator wanted to accept the new data collection as part of the LIMGR, he/she would update the InfoMgr object to create the relationship between the LIMGR and the new data collection.

The LIMGR Configuration and Setup Tool is used to set the options for the LIMGR. In addition to the option for automatic or manual incorporation of new collections, the LIMGR Configuration and Setup Tool can be used to manually force a reread of the DDICT service. This would be used in the manual incorporation mode, when the operator wanted the changes to the DDICT service to take effect in the LIMGR. Another function of the LIMGR Configuration and Setup Tool is to define run-time parameters such as maximum number of sessions, result set memory cache per session, etc.

This page intentionally left blank.

7 GTWAY - Version 0 Gateway CSCI

7.1 CSCI Overview

Gateway provides interoperability with V0 for directory queries, inventory queries, browse requests and product orders. Version 0 queries originating from Version 0 clients will be sent to a Version 0 gateway which will operate at each DAAC. The gateway will translate a incoming V0 ODL request into ECS query format and submit it to the local ECS data server. The result will be returned to the Gateway, which then will reformat it into V0 ODL structures and return it to V0 client. The structure of the V0 ODL messages is documented in "Messages and Development Data Dictionary for v5.0 of IMS Client" (IMSV0-PD-SD-002 v1.0.14 950928). The Gateway uses a database constructed by a gateway administrator using the V0 search parameters, ECS schema and metadata. The Advertising Service (ADSRV) CSCI and Document Data Server (DDSRV) CSCI make use of the gateway database to resolve ECS to V0 mappings.

In Release B, the Gateway becomes bi-directional, meaning that the Gateway also accepts ECS queries and translates them into V0 queries. The V0 queries are sent to the V0 IMS Servers located at the DAACs. This allows the ECS client subsystem to search and order V0 data.

Since this CSCI is on the incremental development track, requirements, schedule, scenarios, issues and design are documented in a Software Development File (SDF) for GTWAY. The GTWAY CSCI requirements are the best understood of all the incremental development track components, thus there is more internal design available in this document for this CSCI.

7.2 Version 0 Gateway Context

Figure 7.2-1 shows the context diagram for the GTWAY CSCI. Table 3.2-1 defines the flows on this diagram.

7.3 Version 0 Gateway Object Model

The GTWAY CSCI has been separated into five categories:

- Request Processing
- Data Server Interface
- Persistent Data
- V0-ECS Mapper
- ECS to V0 Server

The Request Processing object model (Figure 7.3-1) represents the classes containing the requests serviced by the GTWAY CSCI and the corresponding result classes. Each V0 request is converted into an ECS request and is submitted to the data server using data server interface classes described in Section 7.3.2. The result set received from the data server is translated into V0 ODL results message. Requests can be of four types: directory search, inventory search, browse request and product request. A directory search is resolved using the gateway persistent classes described in Section 7.3.3. The result set for directory search contains the GCMD entry ID, the data collection name and the center from which it originated, all of which are stored in the gateway persistent

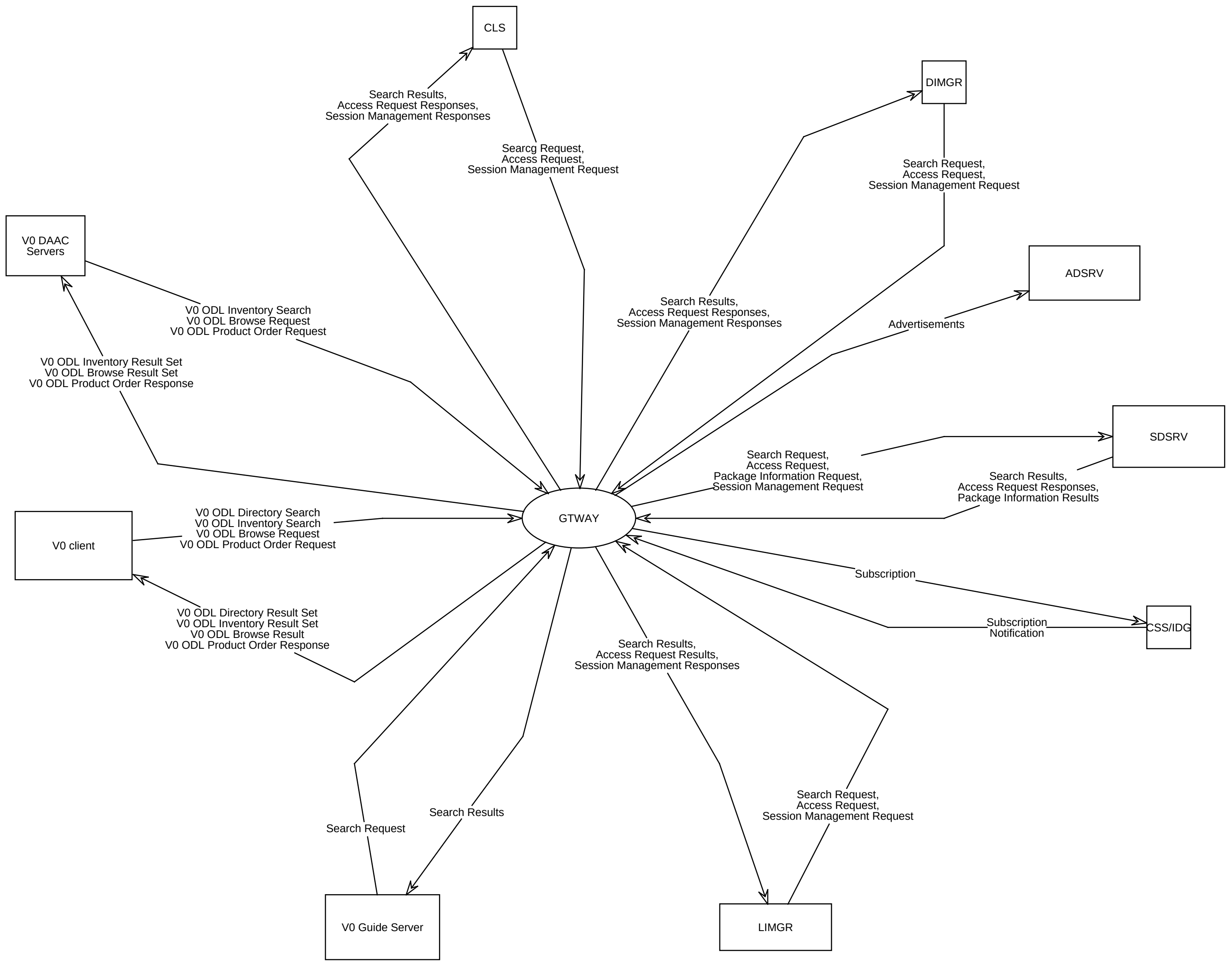

CLS
Search Results,
Access Request Responses,
Session Management Responses
Searcg Request,
Access Request,
Session Management Request
DIMGR
Search Request,
Access Request,
Session Management Request
Search Results,
Access Request Responses,
Session Management Responses
ADSRV
Advertisements
V0 DAAC Servers
V0 ODL Inventory Search
V0 ODL Browse Request
V0 ODL Product Order Request
V0 ODL Inventory Result Set
V0 ODL Browse Result Set
V0 ODL Product Order Response
SDSRV
Search Request,
Access Request,
Package Information Request,
Session Management Request
Search Results,
Access Request Responses,
Package Information Results
V0 client
V0 ODL Directory Search
V0 ODL Inventory Search
V0 ODL Browse Request
V0 ODL Product Order Request
GTWAY
Subscription
Subscription Notification
CSS/IDG
V0 ODL Directory Result Set
V0 ODL Inventory Result Set
V0 ODL Browse Result
V0 ODL Product Order Response
Search Results,
Access Request Results,
Session Management Responses
Search Request,
Access Request,
Session Management Request
Search Results
Search Request
V0 Guide Server
LIMGR

Figure 7.2-1. GTWAY CSCI Context Diagram

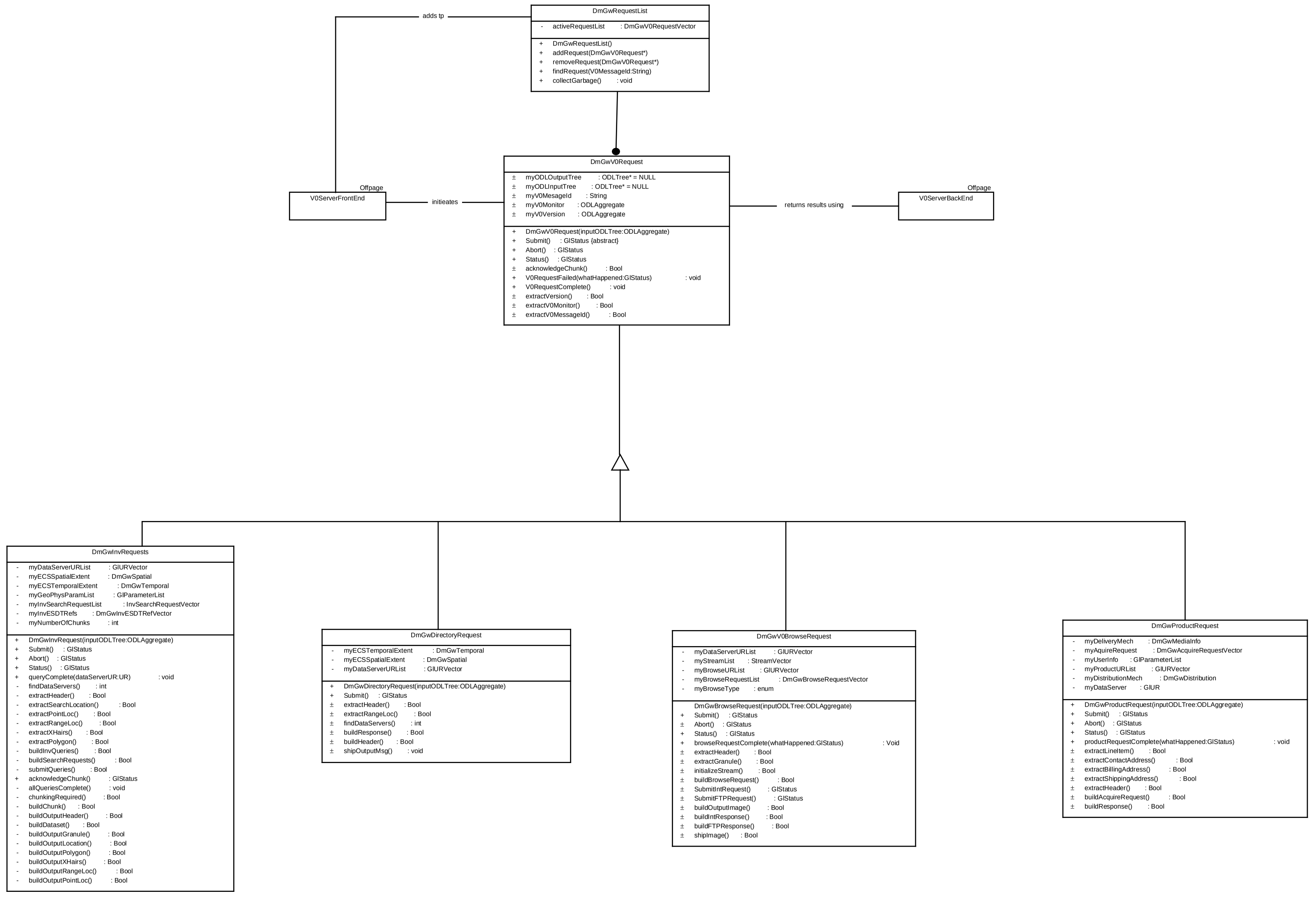
objects. An inventory search request is resolved partially by the gateway using the valids stored in the gateway persistent objects i.e., the gateway identifies the data collection names from the given geophysical parameters, field campaign names, sensor names and satellite names. For each data collection identified, a separate request is formulated to submit to the data server. Data Server sends the results set for each data collection back to the gateway and the gateway formulates the ODL response from the ECS result set. Gateway receives the browse request with the granule id. Gateway persistent objects keep track of the mapping between the Universal Reference (UR) and the granule ID of each granule that was returned to the V0 Client for a certain period. The granule id is translated to a Universal Reference (UR) before submitting the request to the data server. If a browse product is available for direct viewing on the users desktop, the data server returns the browse product with the result set, or it sends a response that the request is being processed for FTP. Product Order requests are translated into ECS data distribution requests and submitted to the data server. The notification received from the data server is translated by the gateway to an ODL response and sent to the V0 client.

The Data Server Interface object model (Figure 7.3-2) represents the interface classes for the Gateway Subsystem to the Data Server Subsystem. For each service request directed from the Request Processing Model, this model provides a service interface. It consists of the objects for handling Inventory Search Requests, Browse Requests, Acquisition Requests, Valid and Distribution Media Requests. The interface objects communicate with the designated data server to submit the service requests. When the service requests are completed, the integrated result and status information are returned back to the Request Processing Module for further processing.

V0 Gateway database stores three types of information: Valids, V0-ECS mapping tables and request tracking. The Persistent Data object model (Figure 7.3-3) includes all the above. Whenever a new ESDT is created by the data server it exports that data type and its valids information. This export information, after being mapped using V0 ECS mapping service is stored in valids persistent module i.e., DmGwDataCollection in its part classes. When the V0 gateway request processing module receives a request for directory search, it resolves the query, using DmGwDataCollection and its part classes. All the required information for resolving the query is stored in the local database. The V0 gateway request processing module resolves the inventory search messages partly using the valids persistent module and determines the list of data collection names to send to the ECS data server. It uses the V0 ECS mapping service to change any of the terms that are being sent to the data server. When the inventory results are returned from the ECS data server, V0 gateway mapping service keeps track of the granule id and the UR relationship using DmGwGranuleIdURMap class. This data is mainly used when the client requests the browse product or tries to order a granule using the granule id. These mappings will be available in the database for a certain period. V0 ECS mapping service maps the ECS terms to V0 terms and vice versa using the contents of the subclasses of DmGwMap. DmGwV0Requests class is intended to be used for recovering the state in case of failures. Current V0 client is not built to recover to the pervious state, but this capability is included to meet the requirements in case that should change in future.

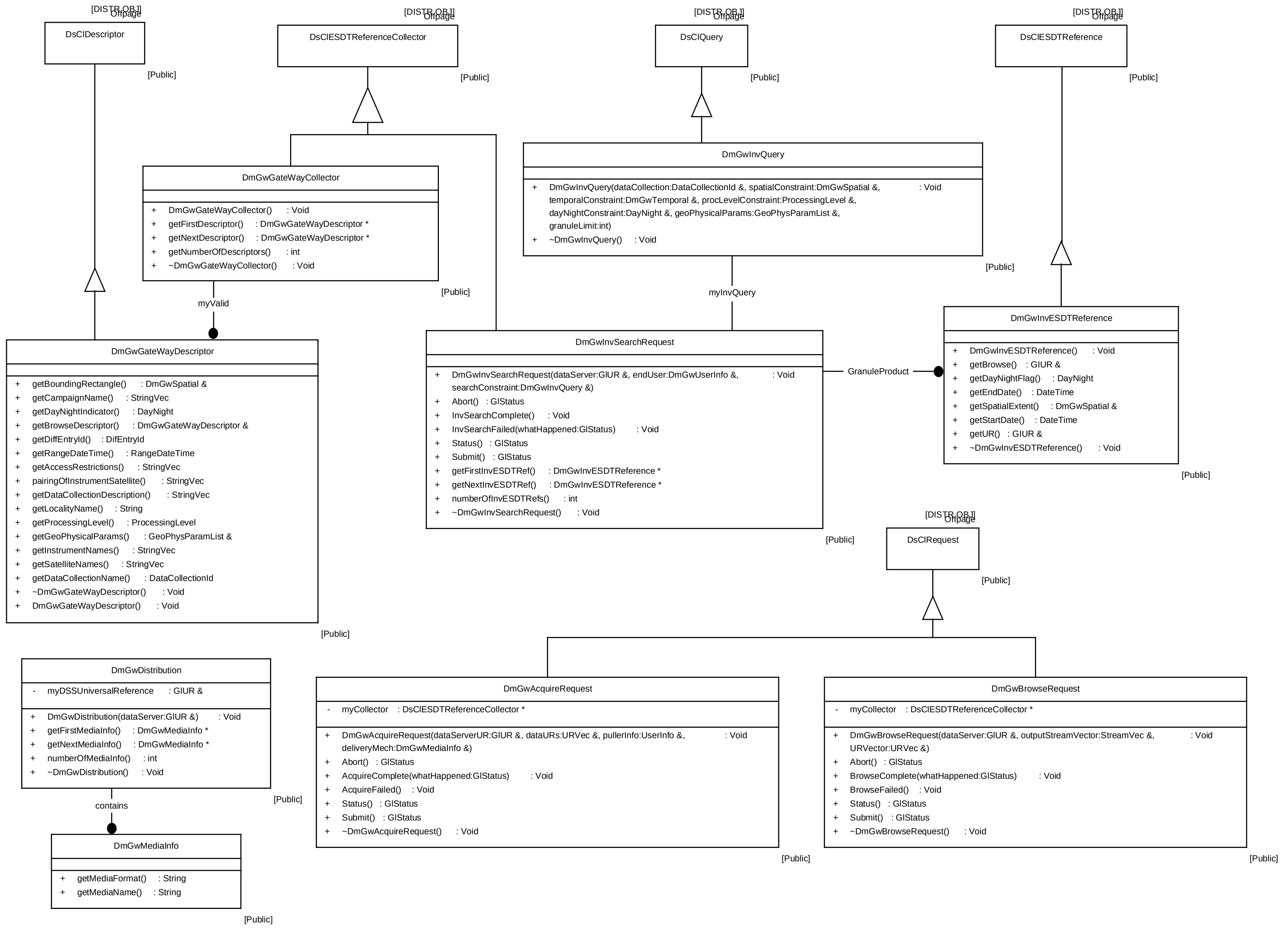
V0 ECS Mapping Service provides the mappings between V0 and ECS schema and the valids. The V0-ECS Mapper object model (Figure 7.3-4) represents the classes and operations required to perform the necessary mapping. Some V0 attributes need to be converted into the ECS domain using a function. For example, a polygon search area specification or a temporal range specification needs conversion to ECS spatial extent specification and temporal extent specification. Certain attributes such as geophysical parameters need domain mapping between the V0 terms and the ECS terms. These domain mappings are available to the mapping service through the gateway persistent objects. When the data server performs the schema export, gateway uses the mapping service to map the ECS terms to V0 terms

The ECS to V0 Server object model (Figure 7.3-5) shows how the Gateway processes requests from an ECS client and sends the request to a V0 IMS server. The client interface to the Gateway is the same as the public interface to the DIMGR. This object model shows the server side objects to the client proxies shown in Figure 5.3.1-1. These objects are subclassed to provide the functionality necessary to interact with V0 IMS servers. The requests are subclassed into those supported by V0, namely Inventory Search, Browse, and Product Request. There are no dynamic service requests with V0, so these are the only types of requests the Gateway has to process. When an ECS request comes into the Gateway, the appropriate V0 request is created and the subclasses such as DmGwV0InvSearchRequest, use the V0 client code to communicate to the V0 server.

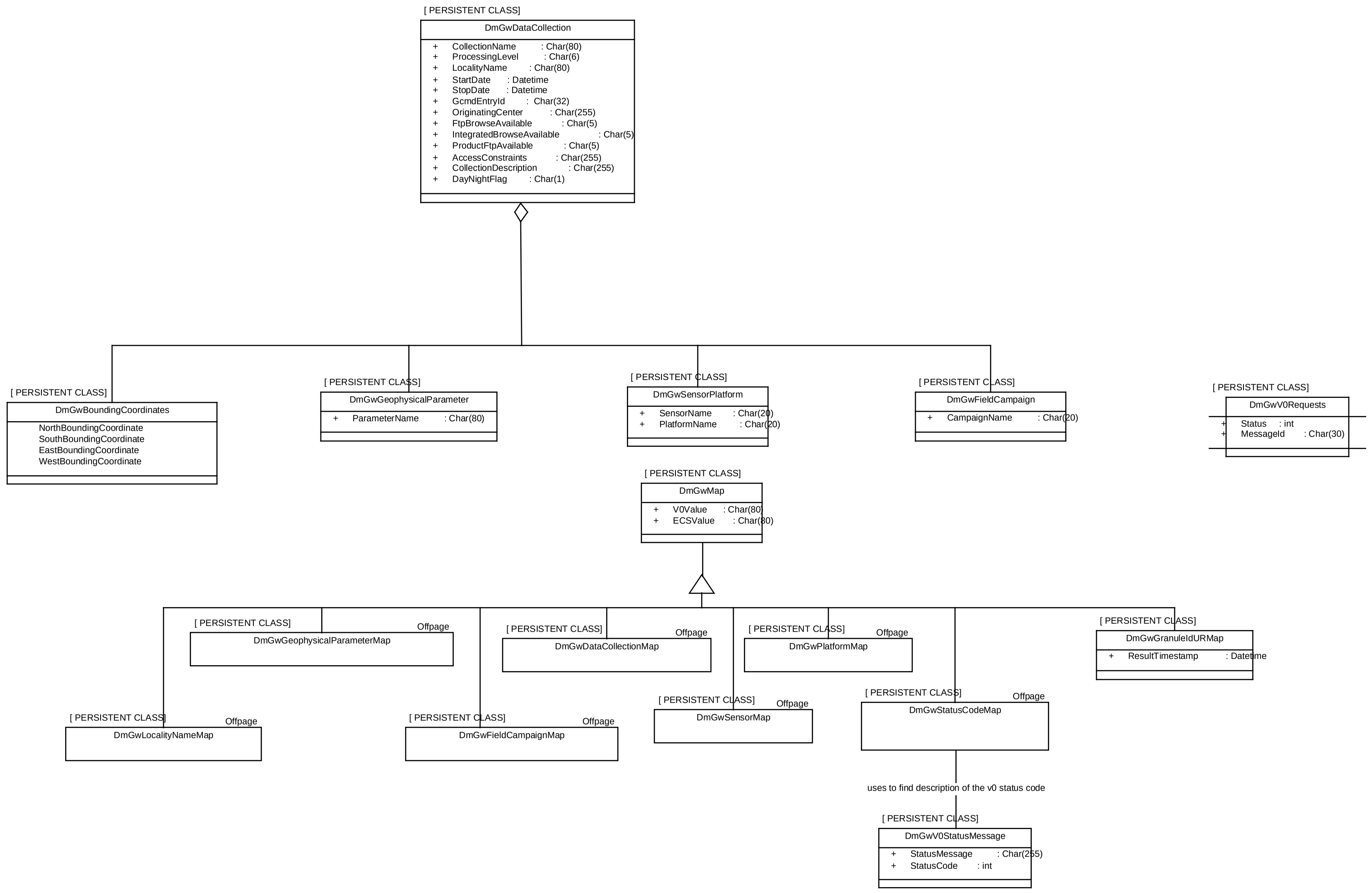


DmGwRequestList
- activeRequestList : DmGwV0RequestVector
+ DmGwRequestList()
+ addRequest(DmGwV0Request*)
+ removeRequest(DmGwV0Request*)
+ findRequest(V0MessageId:String)
+ collectGarbage() : void
adds tp
Offpage
V0ServerFrontEnd
initieates
DmGwV0Request
± myODLOutputTree : ODLTree* = NULL
± myODLInputTree : ODLTree* = NULL
± myV0MesageId : String
± myV0Monitor : ODLAggregate
± myV0Version : ODLAggregate
+ DmGwV0Request(inputODLTree:ODLAggregate)
+ Submit() : GlStatus {abstract}
+ Abort() : GlStatus
+ Status() : GlStatus
± acknowledgeChunk() : Bool
+ V0RequestFailed(whatHappened:GlStatus) : void
+ V0RequestComplete() : void
± extractVersion() : Bool
± extractV0Monitor() : Bool
± extractV0MessageId() : Bool
returns results using
Offpage
V0ServerBackEnd
DmGwInvRequests
- myDataServerURList : GlURVector
- myECSSpatialExtent : DmGwSpatial
- myECSTemporalExtent : DmGwTemporal
- myGeoPhysParamList : GlParameterList
- myInvSearchRequestList : InvSearchRequestVector
- myInvESDTRefs : DmGwInvESDTRefVector
- myNumberOfChunks : int
+ DmGwInvRequest(inputODLTree:ODLAggregate)
+ Submit() : GlStatus
+ Abort() : GlStatus
+ Status() : GlStatus
+ queryComplete(dataServerUR:UR) : void
- findDataServers() : int
- extractHeader() : Bool
- extractSearchLocation() : Bool
- extractPointLoc() : Bool
- extractRangeLoc() : Bool
- extractXHairs() : Bool
- extractPolygon() : Bool
- buildInvQueries() : Bool
- buildSearchRequests() : Bool
- submitQueries() : Bool
+ acknowledgeChunk() : GlStatus
- allQueriesComplete() : void
- chunkingRequired() : Bool
- buildChunk() : Bool
- buildOutputHeader() : Bool
- buildDataset() : Bool
- buildOutputGranule() : Bool
- buildOutputLocation() : Bool
- buildOutputPolygon() : Bool
- buildOutputXHairs() : Bool
- buildOutputRangeLoc() : Bool
- buildOutputPointLoc() : Bool
DmGwDirectoryRequest
- myECSTemporalExtent : DmGwTemporal
- myECSSpatialExtent : DmGwSpatial
- myDataServerURList : GlURVector
+ DmGwDirectoryRequest(inputODLTree:ODLAggregate)
+ Submit() : GlStatus
± extractHeader() : Bool
± extractRangeLoc() : Bool
± findDataServers() : int
± buildResponse() : Bool
± buildHeader() : Bool
± shipOutputMsg() : void
DmGwV0BrowseRequest
- myDataServerURList : GlURVector
- myStreamList : StreamVector
- myBrowseURList : GlURVector
- myBrowseRequestList : DmGwBrowseRequestVector
- myBrowseType : enum
DmGwBrowseRequest(inputODLTree:ODLAggregate)
+ Submit() : GlStatus
± Abort() : GlStatus
+ Status() : GlStatus
+ browseRequestComplete(whatHappened:GlStatus) : Void
± extractHeader() : Bool
± extractGranule() : Bool
± initializeStream() : Bool
± buildBrowseRequest() : Bool
± SubmitIntRequest() : GlStatus
± SubmitFTPRequest() : GlStatus
± buildOutputImage() : Bool
± buildIntResponse() : Bool
± buildFTPResponse() : Bool
± shipImage() : Bool
DmGwProductRequest
- myDeliveryMech : DmGwMediaInfo
- myAquireRequest : DmGwAcquireRequestVector
- myUserInfo : GlParameterList
- myProductURList : GlURVector
- myDistributionMech : DmGwDistribution
- myDataServer : GlUR
+ DmGwProductRequest(inputODLTree:ODLAggregate)
+ Submit() : GlStatus
+ Abort() : GlStatus
+ Status() : GlStatus
+ productRequestComplete(whatHappened:GlStatus) : void
± extractLineItem() : Bool
± extractContactAddress() : Bool
± extractBillingAddress() : Bool
± extractShippingAddress() : Bool
± extractHeader() : Bool
± buildAcquireRequest() : Bool
± buildResponse() : Bool

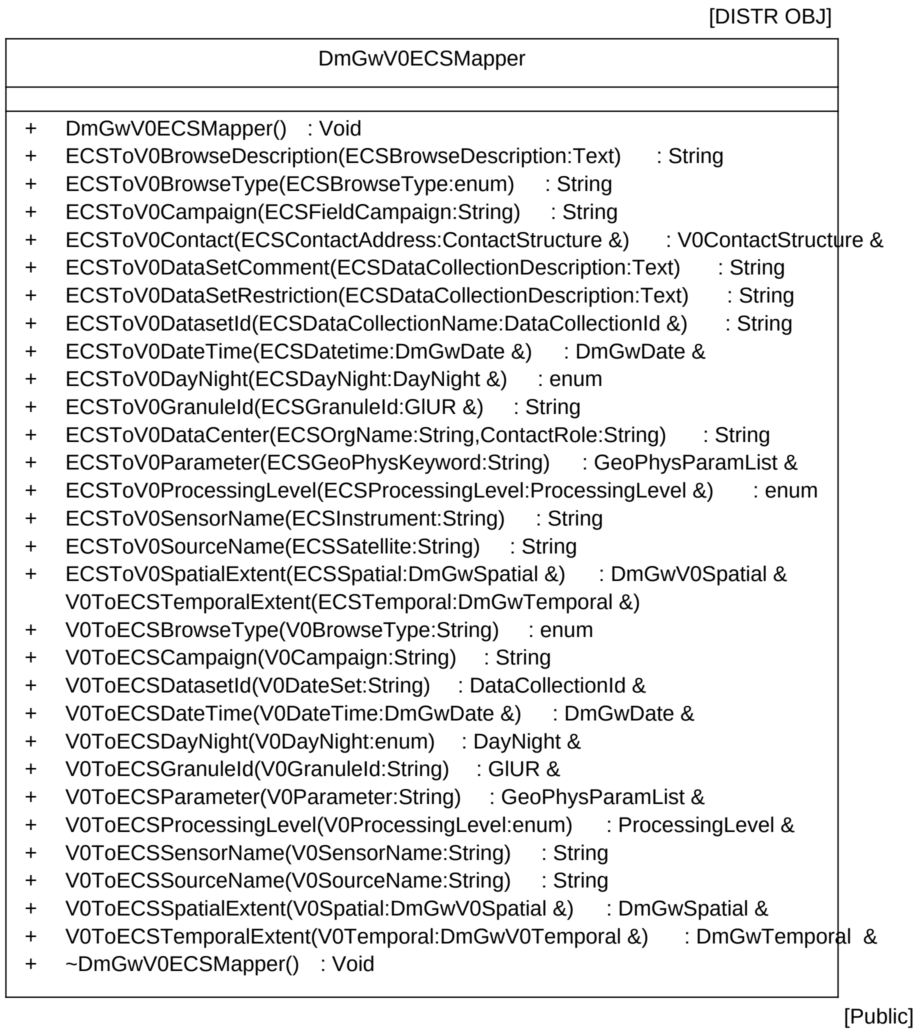
7.3-1. DMGW-Requests Object Model Diagram


[DISTR OBJ] Offpage
DsClDescriptor
[Public]
[DISTR OBJ] Offpage
DsClESDTReferenceCollector
[Public]
[DISTR OBJ] Offpage
DsClQuery
[Public]
[DISTR OBJ] Offpage
DsClESDTReference
[Public]
DmGwGateWayCollector
+ DmGwGateWayCollector() : Void
+ getFirstDescriptor() : DmGwGateWayDescriptor *
+ getNextDescriptor() : DmGwGateWayDescriptor *
+ getNumberOfDescriptors() : int
+ ~DmGwGateWayCollector() : Void
[Public]
myValid
DmGwInvQuery
+ DmGwInvQuery(dataCollection:DataCollectionId &, spatialConstraint:DmGwSpatial &, temporalConstraint:DmGwTemporal &, procLevelConstraint:ProcessingLevel &, dayNightConstraint:DayNight &, geoPhysicalParams:GeoPhysParamList &, granuleLimit:int) : Void
+ ~DmGwInvQuery() : Void
[Public]
myInvQuery
DmGwGateWayDescriptor
+ getBoundingRectangle() : DmGwSpatial &
+ getCampaignName() : StringVec
+ getDayNightIndicator() : DayNight
+ getBrowseDescriptor() : DmGwGateWayDescriptor &
+ getDiffEntryId() : DifEntryId
+ getRangeDateTime() : RangeDateTime
+ getAccessRestrictions() : StringVec
+ pairingOfInstrumentSatellite() : StringVec
+ getDataCollectionDescription() : StringVec
+ getLocalityName() : String
+ getProcessingLevel() : ProcessingLevel
+ getGeoPhysicalParams() : GeoPhysParamList &
+ getInstrumentNames() : StringVec
+ getSatelliteNames() : StringVec
+ getDataCollectionName() : DataCollectionId
+ ~DmGwGateWayDescriptor() : Void
+ DmGwGateWayDescriptor() : Void
[Public]
DmGwInvSearchRequest
+ DmGwInvSearchRequest(dataServer:GIUR &, endUser:DmGwUserInfo &, searchConstraint:DmGwInvQuery &) : Void
+ Abort() : GlStatus
+ InvSearchComplete() : Void
+ InvSearchFailed(whatHappened:GlStatus) : Void
+ Status() : GlStatus
+ Submit() : GlStatus
+ getFirstInvESDTRef() : DmGwInvESDTReference *
+ getNextInvESDTRef() : DmGwInvESDTReference *
+ numberOfInvESDTRefs() : int
+ ~DmGwInvSearchRequest() : Void
[Public]
GranuleProduct
DmGwInvESDTReference
+ DmGwInvESDTReference() : Void
+ getBrowse() : GIUR &
+ getDayNightFlag() : DayNight
+ getEndDate() : DateTime
+ getSpatialExtent() : DmGwSpatial &
+ getStartDate() : DateTime
+ getUR() : GIUR &
+ ~DmGwInvESDTReference() : Void
[Public]
[DISTR OBJ] Offpage
DsClRequest
[Public]
DmGwDistribution
- myDSSUniversalReference : GIUR &
+ DmGwDistribution(dataServer:GIUR &) : Void
+ getFirstMediaInfo() : DmGwMediaInfo *
+ getNextMediaInfo() : DmGwMediaInfo *
+ numberOfMediaInfo() : int
+ ~DmGwDistribution() : Void
[Public]
contains
DmGwMediaInfo
+ getMediaFormat() : String
+ getMediaName() : String
[Public]
DmGwAcquireRequest
- myCollector : DsClESDTReferenceCollector *
+ DmGwAcquireRequest(dataServerUR:GIUR &, dataURs:URVec &, pullerInfo:UserInfo &, deliveryMech:DmGwMediaInfo &) : Void
+ Abort() : GlStatus
+ AcquireComplete(whatHappened:GlStatus) : Void
+ AcquireFailed() : Void
+ Status() : GlStatus
+ Submit() : GlStatus
+ ~DmGwAcquireRequest() : Void
[Public]
DmGwBrowseRequest
- myCollector : DsClESDTReferenceCollector *
+ DmGwBrowseRequest(dataServer:GIUR &, outputStreamVector:StreamVec &, URVector:URVec &) : Void
+ Abort() : GlStatus
+ BrowseComplete(whatHappened:GlStatus) : Void
+ BrowseFailed() : Void
+ Status() : GlStatus
+ Submit() : GlStatus
+ ~DmGwBrowseRequest() : Void
[Public]

7.3-2. DMGWDataServerIF Object Model Diagram



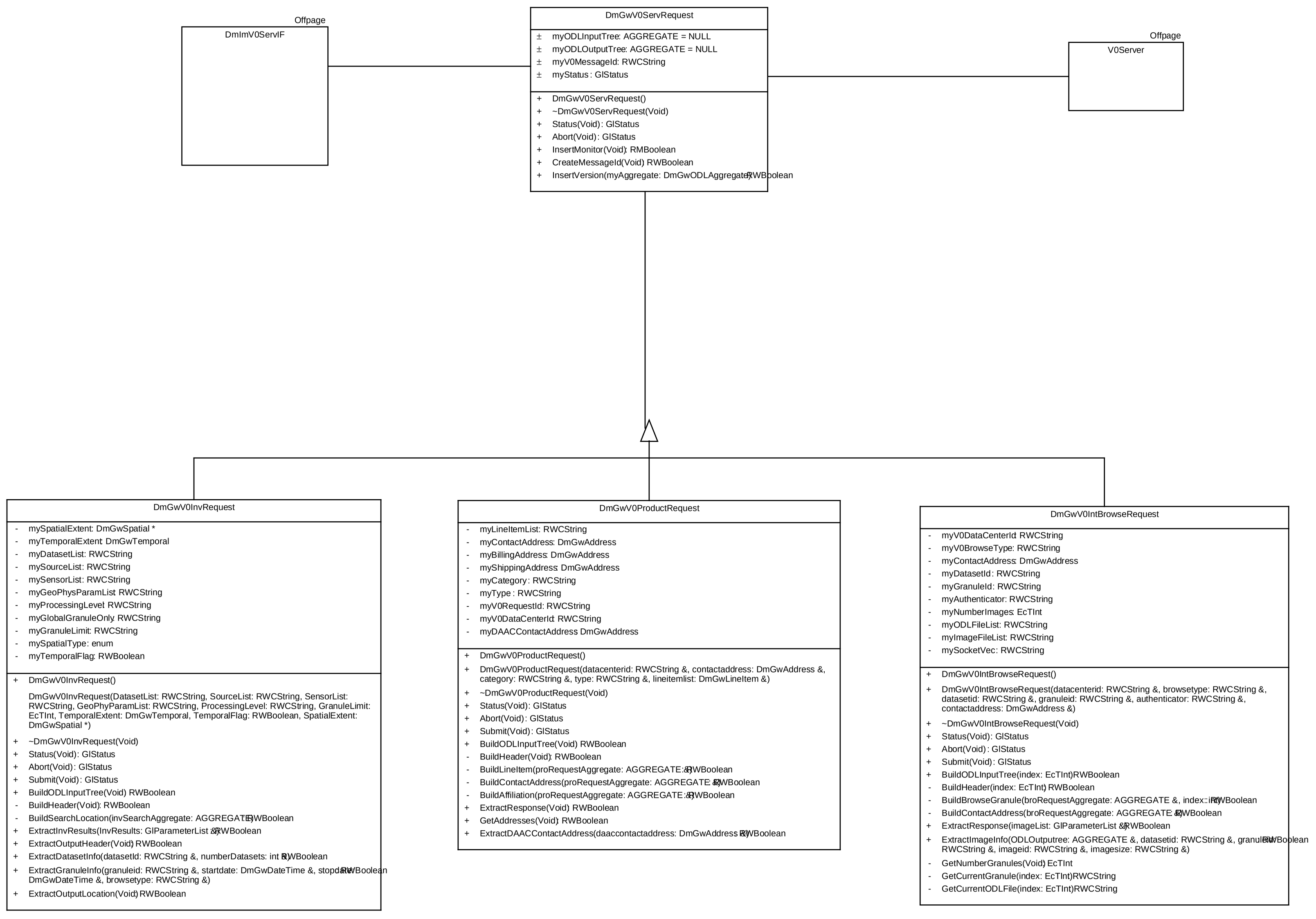
7.3-3. DMGWPersistentData Object Model Diagram



```
[DISTR OBJ]
DmGwV0ECSMapper

+    DmGwV0ECSMapper()   : Void
+    ECSToV0BrowseDescription(ECSBrowseDescription:Text)      : String
+    ECSToV0BrowseType(ECSBrowseType:enum)     : String
+    ECSToV0Campaign(ECSFieldCampaign:String)     : String
+    ECSToV0Contact(ECSContactAddress:ContactStructure &)      : V0ContactStructure &
+    ECSToV0DataSetComment(ECSDataCollectionDescription:Text)      : String
+    ECSToV0DataSetRestriction(ECSDataCollectionDescription:Text)      : String
+    ECSToV0DatasetId(ECSDataCollectionName:DataCollectionId &)      : String
+    ECSToV0DateTime(ECSDatetime:DmGwDate &)     : DmGwDate &
+    ECSToV0DayNight(ECSDayNight:DayNight &)     : enum
+    ECSToV0GranuleId(ECSGranuleId:GlUR &)     : String
+    ECSToV0DataCenter(ECSOrgName:String,ContactRole:String)      : String
+    ECSToV0Parameter(ECSGeoPhysKeyword:String)     : GeoPhysParamList &
+    ECSToV0ProcessingLevel(ECSProcessingLevel:ProcessingLevel &)      : enum
+    ECSToV0SensorName(ECSInstrument:String)     : String
+    ECSToV0SourceName(ECSSatellite:String)     : String
+    ECSToV0SpatialExtent(ECSSpatial:DmGwSpatial &)     : DmGwV0Spatial &
     V0ToECSTemporalExtent(ECSTemporal:DmGwTemporal &)
+    V0ToECSBrowseType(V0BrowseType:String)     : enum
+    V0ToECSCampaign(V0Campaign:String)     : String
+    V0ToECSDatasetId(V0DateSet:String)     : DataCollectionId &
+    V0ToECSDateTime(V0DateTime:DmGwDate &)     : DmGwDate &
+    V0ToECSDayNight(V0DayNight:enum)     : DayNight &
+    V0ToECSGranuleId(V0GranuleId:String)     : GlUR &
+    V0ToECSParameter(V0Parameter:String)     : GeoPhysParamList &
+    V0ToECSProcessingLevel(V0ProcessingLevel:enum)     : ProcessingLevel &
+    V0ToECSSensorName(V0SensorName:String)     : String
+    V0ToECSSourceName(V0SourceName:String)     : String
+    V0ToECSSpatialExtent(V0Spatial:DmGwV0Spatial &)     : DmGwSpatial &
+    V0ToECSTemporalExtent(V0Temporal:DmGwV0Temporal &)      : DmGwTemporal  &
+    ~DmGwV0ECSMapper()   : Void
[Public]
```

7.3-4. DMGWV0ECSMapper Object Model Diagram



Offpage
DmImV0ServIF
DmGwV0ServRequest
± myODLInputTree: AGGREGATE = NULL
± myODLOutputTree: AGGREGATE = NULL
± myV0MessageId: RWCString
± myStatus : GlStatus
+ DmGwV0ServRequest()
+ ~DmGwV0ServRequest(Void)
+ Status(Void): GlStatus
+ Abort(Void): GlStatus
+ InsertMonitor(Void): RMBoolean
+ CreateMessageId(Void): RWBoolean
+ InsertVersion(myAggregate: DmGwODLAggregate): RWBoolean
Offpage
V0Server
DmGwV0InvRequest
- mySpatialExtent: DmGwSpatial *
- myTemporalExtent: DmGwTemporal
- myDatasetList: RWCString
- mySourceList: RWCString
- mySensorList: RWCString
- myGeoPhysParamList: RWCString
- myProcessingLevel: RWCString
- myGlobalGranuleOnly: RWCString
- myGranuleLimit: RWCString
- mySpatialType: enum
- myTemporalFlag: RWBoolean
+ DmGwV0InvRequest()
DmGwV0InvRequest(DatasetList: RWCString, SourceList: RWCString, SensorList: RWCString, GeoPhyParamList: RWCString, ProcessingLevel: RWCString, GranuleLimit: EcTInt, TemporalExtent: DmGwTemporal, TemporalFlag: RWBoolean, SpatialExtent: DmGwSpatial *)
+ ~DmGwV0InvRequest(Void)
+ Status(Void): GlStatus
+ Abort(Void): GlStatus
+ Submit(Void): GlStatus
+ BuildODLInputTree(Void): RWBoolean
- BuildHeader(Void): RWBoolean
- BuildSearchLocation(invSearchAggregate: AGGREGATE): RWBoolean
+ ExtractInvResults(InvResults: GlParameterList &): RWBoolean
+ ExtractOutputHeader(Void): RWBoolean
+ ExtractDatasetInfo(datasetId: RWCString &, numberDatasets: int &): RWBoolean
+ ExtractGranuleInfo(granuleid: RWCString &, startdate: DmGwDateTime &, stopdate: DmGwDateTime &, browsetype: RWCString &): RWBoolean
+ ExtractOutputLocation(Void): RWBoolean
DmGwV0ProductRequest
- myLineItemList: RWCString
- myContactAddress: DmGwAddress
- myBillingAddress: DmGwAddress
- myShippingAddress: DmGwAddress
- myCategory: RWCString
- myType : RWCString
- myV0RequestId: RWCString
- myV0DataCenterId: RWCString
- myDAACContactAddress: DmGwAddress
+ DmGwV0ProductRequest()
+ DmGwV0ProductRequest(datacenterid: RWCString &, contactaddress: DmGwAddress &, category: RWCString &, type: RWCString &, lineitemlist: DmGwLineItem &)
+ ~DmGwV0ProductRequest(Void)
+ Status(Void): GlStatus
+ Abort(Void): GlStatus
+ Submit(Void): GlStatus
+ BuildODLInputTree(Void): RWBoolean
- BuildHeader(Void): RWBoolean
- BuildLineItem(proRequestAggregate: AGGREGATE &): RWBoolean
- BuildContactAddress(proRequestAggregate: AGGREGATE &): RWBoolean
- BuildAffiliation(proRequestAggregate: AGGREGATE &): RWBoolean
+ ExtractResponse(Void): RWBoolean
+ GetAddresses(Void): RWBoolean
+ ExtractDAACContactAddress(daaccontactaddress: DmGwAddress &): RWBoolean
DmGwV0IntBrowseRequest
- myV0DataCenterId: RWCString
- myV0BrowseType: RWCString
- myContactAddress: DmGwAddress
- myDatasetId: RWCString
- myGranuleId: RWCString
- myAuthenticator: RWCString
- myNumberImages: EcTInt
- myODLFileList: RWCString
- myImageFileList: RWCString
- mySocketVec: RWCString
+ DmGwV0IntBrowseRequest()
+ DmGwV0IntBrowseRequest(datacenterid: RWCString &, browsetype: RWCString &, datasetid: RWCString &, granuleid: RWCString &, authenticator: RWCString &, contactaddress: DmGwAddress &)
+ ~DmGwV0IntBrowseRequest(Void)
+ Status(Void): GlStatus
+ Abort(Void): GlStatus
+ Submit(Void): GlStatus
+ BuildODLInputTree(index: EcTInt): RWBoolean
- BuildHeader(index: EcTInt): RWBoolean
- BuildBrowseGranule(broRequestAggregate: AGGREGATE &, index::int): RWBoolean
- BuildContactAddress(broRequestAggregate: AGGREGATE &): RWBoolean
+ ExtractResponse(imageList: GlParameterList &): RWBoolean
+ ExtractImageInfo(ODLOutputree: AGGREGATE &, datasetid: RWCString &, granuleid: RWCString &, imageid: RWCString &, imagesize: RWCString &): RWBoolean
- GetNumberGranules(Void): EcTInt
- GetCurrentGranule(index: EcTInt): RWCString
- GetCurrentODLFile(index: EcTInt): RWCString

7.3-5. IM_V0Interface Object Model Diagram

7.3.1 DmGwAcquireRequest Class

Parent Class: DsClRequest
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required to submit a product ordering request to the data server.

Attributes:

myCollector - This attribute contains the universal reference to the data server which the data collection being requested is binded to.
Data Type: DsClESDTReferenceCollector *
Privilege: Private
Default Value:

Operations:

Abort - This operation cancels the product acquisition request that is being submitted.
Arguments:
Return Type: GlStatus
Privilege: Public

AcquireComplete - This operation should be invoked by the communication layer after the product acquisition request submitted to the data server is completed successfully.
Arguments: whatHappened:GlStatus
Return Type: Void
Privilege: Public

AcquireFailed - This operation should be invoked by the communication layer if a product acquisition request submitted to the data server can not be completed due to the reason specified in the argument whatHappened.
Arguments:
Return Type: Void
Privilege: Public

DmGwAcquireRequest - This operation constructs a product ordering request.
Arguments: dataServerUR:GlUR &, dataURs:URVec &, pullerInfo:UserInfo &, deliveryMech:DmGwMediaInfo &
Return Type: Void
Privilege: Public

Status - This operation is invoked if a desire to check the status of product acquisition is required.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - This operation is invoked to submit the product acquisition request to the data server.
Arguments:
Return Type: GlStatus
Privilege: Public

~DmGwAcquireRequest - This operation destroys the structure of product ordering request.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwAcquireRequest class has associations with the following classes:
None

7.3.2 DmGwBoundingCoordinates Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a part class of DmGwDataCollection. It contains the bounding coorindates of the spatial coverage of the data collection.

Attributes:

EastBoundingCoordinate - Easternmost longitude of the data collection spatial coverage.

NorthBoundingCoordinate - Northernmost latitude of the data collection spatial coverage.

SouthBoundingCoordinate - Southernmost latitude of the data collection spatial coverage.

WestBoundingCoordinate - Westernmost longitude of the data collection spatial coverage.

Operations:

None

Associations:

The DmGwBoundingCoordinates class has associations with the following classes:
DmGwDataCollection (Aggregation)

7.3.3 DmGwBrowseRequest Class

Parent Class: DsClRequest
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required to submit a browse request to the data server.

Attributes:

myCollector - This attribute contains the universal reference to the data server which the browse image being requested is binded to.
Data Type: DsClESDTReferenceCollector *
Privilege: Private
Default Value:

Operations:

Abort - This operation cancels the browse image request that is being submitted.
Arguments:
Return Type: GlStatus
Privilege: Public

BrowseComplete - This operation should be invoked by the communication layer after a browse request submitted to the data server is completed successfully.
Arguments: whatHappened:GlStatus
Return Type: Void
Privilege: Public

BrowseFailed - This operation should be invoked by the communication layer if a browse request submitted to the data server can not be completed due to the reason specified in the argument whatHappened.
Arguments:
Return Type: Void
Privilege: Public

DmGwBrowseRequest - This operation constructs a browse image request.
Arguments: dataServer:GlUR &, outputStreamVector:StreamVec &, URVector:URVec &
Return Type: Void
Privilege: Public

Status - This operation is invoked if a desire to check the status of browse request is required.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - This operation is invoked to submit the browse image request to the data server.
Arguments:
Return Type: GlStatus
Privilege: Public

~DmGwBrowseRequest - This operation destroys the structure of browse image request.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwBrowseRequest class has associations with the following classes:
None

7.3.4 DmGwDataCollection Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the metadata about the data collection stored in the data server associated with V0 gateway. The information in this class and its part classes is used to resolve a V0 directory query and to build the V0 valids file.

Attributes:

AccessConstraints - Descriptive notes about any access restrictions on the data collection.
Data Type: Char(255)
Privilege: Public
Default Value:

CollectionDescription - Brief description of the data collection.
Data Type: Char(255)
Privilege: Public
Default Value:

CollectionName - Name of the data collection which is the same as V0 DATASET_ID.
Data Type: Char(80)
Privilege: Public
Default Value:

DayNightFlag - This flag indicates whether or not the data collection is completely either day or night.
Data Type: Char(1)
Privilege: Public
Default Value:
Contraints:'D', or 'N'
Non Persisent Flag:False

FtpBrowseAvailable - It indicates if browse product for the data collection is available through FTP or not. It is the same as V0 BROWSE, FTP attribute.
Data Type: Char(5)
Privilege: Public
Default Value:

GcmdEntryId - The id assigned by the Global Change Master Directory (GCMD) for the data collection. It is equivalent to V0 attribute MD_ENTRY_ID.
Data Type: Char(32)
Privilege: Public
Default Value:

IntegratedBrowseAvailable - It indicates if the browse product can be viewed through the client. It is the same as V0 BROWSE, INTEGRATED attribute.
Data Type: Char(5)
Privilege: Public
Default Value:

LocalityName - This attribute provides a name denoting the spatial coverage of the data collection. It is equivalent to V0 DATASET_COVERAGE, SPATIAL.
Data Type: Char(80)
Privilege: Public
Default Value:

OriginatingCenter - The data center from where DIF for the data collection has originated. This attribute is used by V0 for GCMD search. It is equivalent to the V0 ORG_CENTER attribute.
Data Type: Char(255)
Privilege: Public
Default Value:

ProcessingLevel - This attribute reflects the classification of the science data processing level, which defines in general terms the characteristics of the output of the processing performed. This is equivalent to V0 PROCESSING_LEVEL attribute.
Data Type: Char(6)
Privilege: Public
Default Value:

ProductFtpAvailable - It indicates whether or not the data collection is available through FTP. It is the same as V0 FTP_PRODUCT_AVAILABLE.
Data Type: Char(5)
Privilege: Public
Default Value:

StartDate - This attribute is the beginning date and time of the data collection temporal coverage.
Data Type: Datetime
Privilege: Public
Default Value:

StopDate - It defines the ending date and time of the data collection temporal coverage.
Data Type: Datetime
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwDataCollection class has associations with the following classes:
None

7.3.5 DmGwDataCollectionMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the mappings, if any, between the dataset names of V0 clients and ECS data server.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwDataCollectionMap class has associations with the following classes:
None

7.3.6 DmGwDirectoryRequest Class

Parent Class: DmGwV0Request
Public: No
Distributed Object:No
Purpose and Description:
The DmGwDirectoryRequest class is a specialization of the DmGwV0Request class. This class is responsible for processing a V0 Directory Search request.

Attributes:

myDataServerURList - stores the list of data server GlURs that are returned from the directory search.
Data Type: GlURVector
Privilege: Private
Default Value:

myECSSpatialExtent - stores the spatial extent to be used in the directory search.
Data Type: DmGwSpatial
Privilege: Private
Default Value:

myECSTemporalExtent - stores the temporal range to be used in the directory search.
Data Type: DmGwTemporal
Privilege: Private
Default Value:

Operations:

DmGwDirectoryRequest - The DmGwDirectoryRequest constructor initializes the myECSTemporalExtent and myECSSpatialExtent attributes from information contained in myODLInputTree.
Arguments: inputODLTree:ODLAggregate
Return Type: Void
Privilege: Public

Submit - The Submit member function is a specializatio of the DmGwV0Request base class member function. It extracts the search constraints from myODLInputTree, performs a directory search on the gateway database, and formats the search output into myODLOutputTree.
Arguments:
Return Type: GlStatus
Privilege: Public

buildHeader - The buildHeader member function builds the output header in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Protected

buildResponse - The buildResponse member functions inserts the directory search results into myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractHeader - The extractHeader member function initializes myECSTemporalExtent attribute from information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractRangeLoc - The extractRangeLoc member function initializes the myECSSpatialExtent attribute from the RangeLoc information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

findDataServers - The findDataServers member function queries the gateway database to perform the inventory search.
Arguments:
Return Type: int
Privilege: Protected

shipOutputMsg - The shipOutputMsg member function delivers myODLOutputTree to the V0ServerBackEnd.
Arguments:
Return Type: void
Privilege: Protected

Associations:

The DmGwDirectoryRequest class has associations with the following classes:
None

7.3.7 DmGwDistribution Class

Parent Class: Not Applicable
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required to acquire product distribution and format information.

Attributes:

myDSSUniversalReference - This attribute contains the universal reference to the distribution data server.
Data Type: GlUR &
Privilege: Private
Default Value:

Operations:

DmGwDistribution - This operation constructs a distribution request for the product format information to the distribution data server.
Arguments: dataServer:GlUR &
Return Type: Void
Privilege: Public

getFirstMediaInfo - This operation retrieves the information of first media format available for the product being ordered.
Arguments:
Return Type: DmGwMediaInfo *
Privilege: Public

getNextMediaInfo - This operation retrieves the information of next media format available for the product being ordered.
Arguments:
Return Type: DmGwMediaInfo *
Privilege: Public

numberOfMediaInfo - This operation retrieves the number of different media formats for the product being ordered.
Arguments:
Return Type: int
Privilege: Public

~DmGwDistribution - This operation destroys the structure of the distribution request for the product format information.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwDistribution class has associations with the following classes:
Class: DmGwMediaInfo contains - This association indicates that the DmGwDistribution contains a collection of DmGwMediaInfo.

7.3.8 DmGwFieldCampaign Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This is a part class of DmGwDataCollection. A data collection may be collected using a campaign which is contained in this class. It is used in resolving the directory search from V0 client and validating the inventory search from V0 client.

Attributes:

CampaignName - The name of the field campaign.
Data Type: Char(20)
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwFieldCampaign class has associations with the following classes:
DmGwDataCollection (Aggregation)

7.3.9 DmGwFieldCampaignMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the mapping between V0 field campaign names and the ECS field campaign names.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwFieldCampaignMap class has associations with the following classes:
None

7.3.10 DmGwGateWayCollector Class

Parent Class: DsClESDTReferenceCollector
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required to retrieve a collection of valids exported from the data server.

Attributes:

All Attributes inherited from parent class

Operations:

DmGwGateWayCollector - This operation constructs a valid export request to the data server.
Arguments:
Return Type: Void
Privilege: Public

getFirstDescriptor - This operation retrieves the information of the first valid descriptor exported from the data server.
Arguments:
Return Type: DmGwGateWayDescriptor *
Privilege: Public

getNextDescriptor - This operation retrieves the information of the next valid descriptor exported from the data server.
Arguments:
Return Type: DmGwGateWayDescriptor *
Privilege: Public

getNumberOfDescriptors - This operation retrieves the number of valid descriptors exported from the data server.
Arguments:
Return Type: int
Privilege: Public

~DmGwGateWayCollector - This operation destroyes the structure of the valid export request.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwGateWayCollector class has associations with the following classes:
Class: DmGwGateWayDescriptor myValid - This association indicates that DmGwGateWayCollector is a collection of DmGwGateWayDescriptor valids.

7.3.11 DmGwGateWayDescriptor Class

Parent Class: DsClDescriptor
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains the valid information which is returned by the data server after the valid export request is completed.

Attributes:

All Attributes inherited from parent class

Operations:

DmGwGateWayDescriptor - This operation constructs a valid descriptor for storing exported valid information.
Arguments:
Return Type: Void
Privilege: Public

getAccessRestrictions - This operation returns the ordering restriction and legal prerequisites placed on the data set.
Arguments:
Return Type: StringVec
Privilege: Public

getBoundingRectangle - This operation returns the specification of the spatial coverage for each data set.
Arguments:
Return Type: DmGwSpatial &
Privilege: Public

getBrowseDescriptor - This operation returns the browse descriptor, if exists, which is related to the data set.
Arguments:
Return Type: DmGwGateWayDescriptor &
Privilege: Public

getCampaignName - This operation returns the name of campaign or project that gathered the data set.
Arguments:
Return Type: StringVec
Privilege: Public

getDataCollectionDescription - This operation returns the major emphasis of the content of the data collection.
Arguments:
Return Type: StringVec
Privilege: Public

getDataCollectionName - This operation returns the recommended name to be used when referring to this data set.
Arguments:
Return Type: DataCollectionId
Privilege: Public

getDayNightIndicator - This operation returns the day/night indicator for the data set.
Arguments:
Return Type: DayNight
Privilege: Public

getDiffEntryId - This operation returns the entry id of the Global Change Master Directory for the data set.
Arguments:
Return Type: DifEntryId
Privilege: Public

getGeoPhysicalParams - This operation returns the specification of the geophysical parameters referenced in the data set.
Arguments:
Return Type: GeoPhysParamList &
Privilege: Public

getInstrumentNames - This operation returns the abbreviation, acronym, or other common name of the instrument sensor by which the data set is collected.
Arguments:
Return Type: StringVec
Privilege: Public

getLocalityName - This operation returns the spacial coverage described for the data set.
Arguments:
Return Type: String
Privilege: Public

getProcessingLevel - This operation returns the classification of the science data processing level which defines the characteristics of the data set.
Arguments:
Return Type: ProcessingLevel
Privilege: Public

getRangeDateTime - This operation returns the temporal coverage period extended for the data set.
Arguments:
Return Type: RangeDateTime
Privilege: Public

getSatelliteNames - This operation returns the name of the satellite on which the data set is collected.
Arguments:
Return Type: StringVec
Privilege: Public

pairingOfInstrumentSatellite - This operation returns the matching pairs for instrument sensor and satellite.
Arguments:
Return Type: StringVec
Privilege: Public

~DmGwGateWayDescriptor - This operation destroys the structure of valid descriptor.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwGateWayDescriptor class has associations with the following classes:
Class: DmGwGateWayCollector myValid - This association indicates that DmGwGateWayCollector is a collection of DmGwGateWayDescriptor valids.

7.3.12 DmGwGeophysicalParameter Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This is a part class of DmGwDataCollection. It contains the goephysical parameter names that the data collection references.

Attributes:

ParameterName - The geophysical parameter name, such as "Sea Surface Temperature".
Data Type: Char(80)
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwGeophysicalParameter class has associations with the following classes:
DmGwDataCollection (Aggregation)

7.3.13 DmGwGeophysicalParameterMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the mappings between V0 geophysical parameters and ECS geophysical parameters. A single V0 parameter may map to many ECS parameters and vice versa.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwGeophysicalParameterMap class has associations with the following classes:
None

7.3.14 DmGwGranuleIdURMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the UR and the granule id of every result granule of an inventory search. This mapping is maintained in the V0 gateway database so that any further requests from the V0 client using the granule id can be translated into UR before requesting the data server. These results are stored for certain period of time and then purged out.

Attributes:

ResultTimestamp - Timestamp when the result was returned to the V0 client. It is used in purging the mapping 72 hours after it has been inserted.
Data Type: Datetime
Privilege: Public
Default Value:
Contraints:
Non Persisent Flag:False

Operations:

All Operations inherited from parent class

Associations:

The DmGwGranuleIdURMap class has associations with the following classes:
None

7.3.15 DmGwInvESDTReference Class

Parent Class: DsClESDTReference
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains the information of granule references returned from an inventory search request for a particular dataset.

Attributes:

All Attributes inherited from parent class

Operations:

DmGwInvESDTReference - This operation constructs the structure for storing granule related information.
Arguments:
Return Type: Void
Privilege: Public

getBrowse - This operation retrieves the browse reference, if exists, which is related to the individual granule returned in the inventory data set.
Arguments:
Return Type: GIUR &
Privilege: Public

getDayNightFlag - This operation retrieves the day/night indication of individual granule returned for the inventory data set.
Arguments:
Return Type: DayNight
Privilege: Public

getEndDate - This operation retrieves the ending time of the temporal coverage for the individual granule returned in the inventory data set.
Arguments:
Return Type: DateTime
Privilege: Public

getSpatialExtent - This operation retrieves the spatial coverage of individual granule returned in the inventory data set.
Arguments:
Return Type: DmGwSpatial &
Privilege: Public

getStartDate - This operation retrieves the starting time of the temporal coverage for the individual granule returned in the inventory data set.
Arguments:
Return Type: DateTime
Privilege: Public

getUR - This operation returns the universal reference to the individual granule returned in the inventory data set.
Arguments:
Return Type: GIUR &
Privilege: Public

~DmGwInvESDTReference - This operation destroys the granule reference structure.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwInvESDTReference class has associations with the following classes:

Class: DmGwInvSearchRequest GranuleProduct - This association indicates that the DmGwInvESDTReference is a collection of granule product returned from the inventory search request DmGwInvSearchRequest.

7.3.16 DmGwInvQuery Class

Parent Class: DsClQuery
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains the information of the query criteria for the data set requested.

Attributes:

All Attributes inherited from parent class

Operations:

DmGwInvQuery - This class contains the information of the query criteria for the data set requested.
Arguments: dataCollection:DataCollectionId &, spatialConstraint:DmGwSpatial &, temporalConstraint:DmGwTemporal &, procLevelConstraint:ProcessingLevel &, dayNightConstraint:DayNight &, geoPhysicalParams:GeoPhysParamList &, granuleLimit:int
Return Type: Void
Privilege: Public

~DmGwInvQuery - This operation destroys the query structure for the data set.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwInvQuery class has associations with the following classes:

Class: DmGwInvSearchRequest myInvQuery - This association indicates that DmGwInvQuery is the query criteria when the inventory search request DmGwInvSearchRequest is submitted.

7.3.17 DmGwInvRequests Class

Parent Class: DmGwV0Request
Public: No
Distributed Object:No
Purpose and Description:
The DmGwInvRequest class is a specialization of the DmGwV0Request class. This class is responsible for processing a V0 Inventory Search Request.

Attributes:

myDataServerURList - list of GlUR pointers for the data servers to be searched.
Data Type: GlURVector
Privilege: Private
Default Value:

myECSSpatialExtent - spatial constraint specified for the inventory search.
Data Type: DmGwSpatial
Privilege: Private
Default Value:

myECSTemporalExtent - temporal constraint specified for the inventory search.
Data Type: DmGwTemporal
Privilege: Private
Default Value:

myGeoPhysParamList - list of geo-physical parameters specified for the inventory search.
Data Type: GlParameterList
Privilege: Private
Default Value:

myInvESDTRefs - list of DmGwInvESDRReferences that are returned as a result of a data server search.
Data Type: DmGwInvESDTRefVector
Privilege: Private
Default Value:

myInvSearchRequestList - list of DmGwInvSearchRequest pointers. There is one pointer per each data server to be searched.
Data Type: InvSearchRequestVector
Privilege: Private
Default Value:

myNumberOfChunks - stores the number of V0 Chunk responses required.
Data Type: int
Privilege: Private
Default Value:

Operations:

Abort - The Abort member function is a specialization of the DmGwV0Request base class member function. It aborts any outstanding data server search requests and terminates further processing of this inventory search.
Arguments:
Return Type: GlStatus
Privilege: Public

DmGwInvRequest - The DmGwInvRequest constructor.
Arguments: inputODLTree:ODLAggregate
Return Type: Void
Privilege: Public

Status - The Status member function is a specialization of the DmGwV0Request base class member function. It returns the current status of the inventory search being processed.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function is a specialization of the DmGwV0Request base class member function. It extracts the search constraints from myODLInputTree, determines which data servers should be queried, builds search request for each data server, and submits the requests to the data servers.
Arguments:
Return Type: GlStatus
Privilege: Public

acknowledgeChunk - The acknowledgeChunk member function is a specialization of the DmGwV0Request base class member function. It prepares and ships the next response chunk.
Arguments:
Return Type: GlStatus
Privilege: Public

allQueriesComplete - The allQueriesComplete member function prepares the ODL response message in myODLOutputTree. If chunking is required is builds and ships the first chunk only.
Arguments:
Return Type: void
Privilege: Private

buildChunk - The buildChunk member function builds the myODLOutputTree attribute for one chunk of inventory results.
Arguments:
Return Type: Bool
Privilege: Private

buildDataset - The buildDataset member function builds the dataset header in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildInvQueries - The buildInvQueries member function builds a DmGwInvQuery object for each data server to be queried.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputGranule - The buildOutputGranule builds the granule information in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputHeader - The buildOutputHeader builds the response header in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputLocation - The buildOutputLocation builds the granule spatial location in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputPointLoc - The buildOutputPointLoc builds the granule PointLoc spatial location in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputPolygon - The buildOutputPolygon builds the granule Polygon spatial location in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputRangeLoc - The buildOutputLocation builds the granule RangeLoc spatial location in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildOutputXHairs - The buildOutputLocation builds the granule spatial location in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Private

buildSearchRequests - The buildInvQueries member function builds a DmGwInvSearchRequest object for each data server to be queried.
Arguments:
Return Type: Bool
Privilege: Private

chunkingRequired - The chunkingRequired member function returns True if myODLOutputTree is greater than 64K.
Arguments:
Return Type: Bool
Privilege: Private

extractHeader - The extractHeader member function initializes the myGeoPhysParamList and myECSTemporalExtent attributes from information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Private

extractPointLoc - The extractPointLoc member function initializes the myECSSpatialExtent attribute from the pointLoc information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Private

extractPolygon - The extractPolygon member function initializes the myECSSpatialExtent attribute from the polygon information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Private

extractRangeLoc - The extractRangeLoc member function initializes the myECSSpatialExtent attribute from the RangeLoc information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Private

extractSearchLocation - The extractSearchLocation member function determines the type of spatial information that is contained in myODLInputTree. Based on the spatial type, the appropriate extract member function will be called.
Arguments:
Return Type: Bool
Privilege: Private

extractXHairs - The extractXHairs member function initializes the myECSSpatialExtent attribute from the XHairs information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Private

findDataServers - The findDataServers member function queries the gateway database to determine which data servers should be queried for the inventory search.

Arguments:	
Return Type:	int
Privilege:	Private

queryComplete - The queryComplete member function is called by each DmGwInvSearchRequest object when it's data server search has completed.

Arguments:	dataServerUR:UR
Return Type:	void
Privilege:	Public

submitQueries - The submitQueries member function submits each of the DmGwInvSearchRequest objects.

Arguments:	
Return Type:	Bool
Privilege:	Private

Associations:

The DmGwInvRequests class has associations with the following classes:

None

7.3.18 DmGwInvSearchRequest Class

Parent Class: DsClESDTReferenceCollector
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required to submit an inventory search request to the data server.

Attributes:

All Attributes inherited from parent class

Operations:

Abort - This operation cancels the inventory search request that is being submitted.
Arguments:
Return Type: GlStatus
Privilege: Public

DmGwInvSearchRequest - This operation constructs an inventory search request.
Arguments: dataServer:GlUR &, endUser:DmGwUserInfo &, searchConstraint:DmGwInvQuery &
Return Type: Void
Privilege: Public

InvSearchComplete - This operation should be invoked by the communication layer after the inventory search request submitted to the data server is completed successfully.
Arguments:
Return Type: Void
Privilege: Public

InvSearchFailed - This operation should be invoked by the communication layer if an inventory search request submitted to the data server can not be completed due to the reason specified in the argument whatHappened.
Arguments: whatHappened:GlStatus
Return Type: Void
Privilege: Public

Status - This operation is invoked if a desire to check the status of inventory search request is required.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - This operation is invoked to submit the inventory search request to the data server.
Arguments:
Return Type: GlStatus
Privilege: Public

getFirstInvESDTRef - This operation retrieves the information of the first granule reference returned from the inventory search request.
Arguments:
Return Type: DmGwInvESDTReference *
Privilege: Public

getNextInvESDTRef - This operation retrieves the information of the next granule reference returned from an inventory search request.
Arguments:
Return Type: DmGwInvESDTReference *
Privilege: Public

numberOfInvESDTRefs - This operation retrieves the number of granule references returned in the data set requested.
Arguments:
Return Type: int
Privilege: Public

~DmGwInvSearchRequest - This operation destroys the structure of inventory search request.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwInvSearchRequest class has associations with the following classes:
Class: DmGwInvESDTReference GranuleProduct - This association indicates that the DmGwInvESDTReference is a collection of granule product returned from the inventory search request DmGwInvSearchRequest.
Class: DmGwInvQuery myInvQuery - This association indicates that DmGwInvQuery is the query criteria when the inventory search request DmGwInvSearchRequest is submitted.

7.3.19 DmGwLocalityNameMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores mapping from V0 locality names (which are mapped to DATASET_COVERAGE, SPATIAL) to ECS locality names.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwLocalityNameMap class has associations with the following classes:
None

7.3.20 DmGwMap Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class provides the mappings between the V0 terms and the ECS terms. V0 valids such as geophysical parameters, sensors, platforms may have different terminology in ECS for which the mappings are stored using this class. This is an abstract class where term can be of several types and they appear as subclasses of this class.

Attributes:

ECSValue - This is the ECS term.
Data Type: Char(80)
Privilege: Public
Default Value:

V0Value - This is the V0 term.
Data Type: Char(80)
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwMap class has associations with the following classes:
None

7.3.21 DmGwMediaInfo Class

Parent Class: Not Applicable
Public: Yes
Distributed Object:No
Purpose and Description:
This class contains all the information and the operations required for acquiring distribution media format.

Attributes:

None

Operations:

getMediaFormat - This operation retrieves the format of the distribution media.
Arguments:
Return Type: String
Privilege: Public

getMediaName - This operation retrieves the name of the distribution media.
Arguments:
Return Type: String
Privilege: Public

Associations:

The DmGwMediaInfo class has associations with the following classes:
Class: DmGwDistribution contains - This association indicates that the DmGwDistribution contains a collection of DmGwMediaInfo.

7.3.22 DmGwPlatformMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the mappings between V0 platforms and the ECS platforms.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwPlatformMap class has associations with the following classes:
None

7.3.23 DmGwProductRequest Class

Parent Class: DmGwV0Request
Public: No
Distributed Object:No
Purpose and Description:
The DmGwProductRequestClass is a specialization of the DmGwV0Request class. This class is responsible for processing a V0 Product Request (i.e. data order request).

Attributes:

myAquireRequest - stores the DmGwAcquireRequest for the product order.
Data Type: DmGwAcquireRequestVector
Privilege: Private
Default Value:

myDataServer - stores the UR of the data server to be used for the product request.

Data Type:	GlUR
Privilege:	Private
Default Value:	

myDeliveryMech - stores the DmGwMediaInfo type for the product request.

Data Type:	DmGwMediaInfo
Privilege:	Private
Default Value:	

myDistributionMech - stores the data server distribution interface object.

Data Type:	DmGwDistribution
Privilege:	Private
Default Value:	

myProductURList - list of URs for the data items to be ordered.

Data Type:	GlURVector
Privilege:	Private
Default Value:	

myUserInfo - stores the list of user information (e.g. e-mail address) needed for a product order.

Data Type:	GlParameterList
Privilege:	Private
Default Value:	

Operations:

Abort - The Status member function is a specialization of the DmGwV0Request base class member function. It aborts any outstanding distribution requests and terminates further processing of this product request.

Arguments:	
Return Type:	GlStatus
Privilege:	Public

DmGwProductRequest - The DmGwProductRequest constructor.

Arguments:	inputODLTree:ODLAggregate
Return Type:	Void
Privilege:	Public

Status - The Status member function is a specialization of the DmGwV0Request base class member function. It returns the current status of the inventory search being processed.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function is a specialization of the DmGwV0Request base class member function. It uses myDistributionMech to order the product data.
Arguments:
Return Type: GlStatus
Privilege: Public

buildAcquireRequest - The buildAcquireRequest member function initializes the myAcquireRequest attribute.
Arguments:
Return Type: Bool
Privilege: Protected

buildResponse - The buildResponse member function is responsible for building the V0 response message in myODLOutputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractBillingAddress - The extractBillingAddress member function initializes a portion of myUserInfo from myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractContactAddress - The extractContactAddress member function initializes portions of myUserInfo from myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractHeader - The extractHeader member function controls the initialization of class attributes from information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractLineItem - The extractLineItem member function extracts the granule order data from the LineItem portion of myODLIntputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractShippingAddress - The extractShippingAddress member function initializes a portion of myUserInfo from myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

productRequestComplete - The productRequestComplete member function is called by the DmGwACquireRequest object when the acquire operation is complete.
Arguments: whatHappened:GlStatus
Return Type: void
Privilege: Public

Associations:

The DmGwProductRequest class has associations with the following classes:
None

7.3.24 DmGwRequestList Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The DmGwRequestList class is a container object that maintains the list of currently active V0 Requests. Each V0 Request is represented by a DmGwV0Request object.

Attributes:

activeRequestList - stores the collection of active DmGwV0Request objects.
Data Type: DmGwV0RequestVector
Privilege: Private
Default Value:

Operations:

DmGwRequestList - The DmGwRequestList constructor initializes the activeRequestList.

Arguments:	
Return Type:	Void
Privilege:	Public

addRequest - The addRequest member function adds a new DmGwV0Request object to the activeRequestList.

Arguments:	DmGwV0Request*
Return Type:	Void
Privilege:	Public

collectGarbage - The collectGarbage member function is responsible for the removal and deletion of DmGwV0Request objects. Whenever a DmGwV0Request object has completed processing, it registers itself for garbage collection by calling the removeRequest member function of the DmGwRequestList object. The request list object then schedules a garbage collection callback which initiates the collectGarbage member function.

Arguments:	
Return Type:	void
Privilege:	Public

findRequest - The findRequest member function returns the address of a DmGwV0Request object with the V0MessageId specified in the argument. An object pointer will only be returned if it is currently on teh activeREquestList.

Arguments:	V0MessageId:String
Return Type:	Void
Privilege:	Public

removeRequest - The removeRequest member function removes a DmGwV0Request from the activeRequestList and schedules the deletion of the DmGwV0Request via the collectGarbage member function.

Arguments:	DmGwV0Request*
Return Type:	Void
Privilege:	Public

Associations:

The DmGwRequestList class has associations with the following classes:

Class: DmGwV0Request

Class: V0ServerFrontEnd addstp - The V0ServerFrontEnd adds requests to the DmGwRequestList.

7.3.25 DmGwSensorMap Class

Parent Class: DmGwMap

Public: No

Distributed Object:No

Persistent Class:True

Purpose and Description:

This class stores the mappings between the V0 sensor names and ECS sensor names

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwSensorMap class has associations with the following classes:

None

7.3.26 DmGwSensorPlatform Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This is a part class of the DmGwDataCollection. It contains the sensor names used for measurement of the data collection and its associated platform names. This class is used in resolving V0 directory search requests and validating V0 inventory queries.

Attributes:

PlatformName - The platform or satellite name.
Data Type: Char(20)
Privilege: Public
Default Value:

SensorName - The sensor or instrument name.
Data Type: Char(20)
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwSensorPlatform class has associations with the following classes:
DmGwDataCollection (Aggregation)

7.3.27 DmGwStatusCodeMap Class

Parent Class: DmGwMap
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the mappings between ECS status codes and V0 status codes. Many ECS status codes may map to a single V0 status code and vice versa

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DmGwStatusCodeMap class has associations with the following classes:
Class: DmGwV0StatusMessage usestofinddescriptionofthev0statuscode

7.3.28 DmGwV0BrowseRequest Class

Parent Class: DmGwV0Request
Parent Class: DmGwV0ServRequest
Public: No
Distributed Object:No
Purpose and Description:
The DmGwV0BrowseRequest is a specialization of the DmGwV0Request class. This class is responsible for processing A V0 Browse request.

Attributes:

myBrowseRequestList - stores the list of DmGwBrowseRequest objects that are processing the individual browse requests.
Data Type: DmGwBrowseRequestVector
Privilege: Private
Default Value:

myBrowseType - stores the type of browse acquisition. Possible types are integrated and FTP-Pull.
Data Type: enum
Privilege: Private
Default Value:

myBrowseURList - stores the list of GlURs which point to the browse data to be obtained.
Data Type: GlURVector
Privilege: Private
Default Value:

myDataServerURList - stores the list of GlURs for the data servers which will process the browse requests.
Data Type: GlURVector
Privilege: Private
Default Value:

myStreamList - stores the list of stream objects which will receive the browse data from the data server.
Data Type: StreamVector
Privilege: Private
Default Value:

Operations:

Abort - The Abort member function is a specialization of the DmGwV0Request base class member function. It aborts any outstanding DmGwBrowseRequest objects in myBrowseRequestList and terminates further processing of this V0 Browse Request.
Arguments:
Return Type: GlStatus
Privilege: Protected

DmGwBrowseRequest - The DmGWBrowseRequest constructor initializes the class attribute set from informatio contained in myODLInputTree.
Arguments: inputODLTree:ODLAggregate

Status - The Status member function is a specialization of the DmGwV0Request base class member function. It returns the current status of the browse request being processed.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function is a specialization of the DmGwV0Request base class member function. It submits the browse request by invoking either SubmitIntRequest or SubmitFTPRequest.
Arguments:
Return Type: GlStatus
Privilege: Public

SubmitFTPRequest - The SubmitFtpRequest submits an FTP-Pull browse request to the data server via myBrowseRequestList.
Arguments:
Return Type: GlStatus
Privilege: Protected

SubmitIntRequest - The SubmitIntRequest submits an integrated browse request to the data server via myBrowseRequestList.
Arguments:
Return Type: GlStatus
Privilege: Protected

browseRequestComplete - The browseRequestComplete member function is called by each of the DmGwBroseRequest objects in myBrowseRequestList as they complete the browse request to the data server.
Arguments: whatHappened:GlStatus
Return Type: Void
Privilege: Public

buildBrowseRequest - The buildBrowseRequest member function initializes each object in myBrowseRequestList.
Arguments:
Return Type: Bool
Privilege: Protected

buildFTPResponse - The buildFTPResponse member function builds the myODLOutputTree attribute for an FTP-Pull browse response message.
Arguments:
Return Type: Bool
Privilege: Protected

buildIntResponse - The buildIntResponse member function builds the myODLOutputTree attribute for an integrated browse response message.
Arguments:
Return Type: Bool
Privilege: Protected

buildOutputImage - The buildOutputImage member function delivers the image data from myStreamList to the V0ServerBackEnd object.
Arguments:
Return Type: Bool
Privilege: Protected

extractGranule - the extractGranule member function initializes the myBroseURList from information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

extractHeader - The extractHeader member function initializes the myBrowseType attribute from information contained in myODLInputTree.
Arguments:
Return Type: Bool
Privilege: Protected

initializeStream - The initializeStream member function initializes each of the stream objects in myStreamList.
Arguments:
Return Type: Bool
Privilege: Protected

shipImage - The shipImage member function delivers the myODLOutputTree attribute to the V0ServerBackEnd for transmission to the V0 Client.
Arguments:
Return Type: Bool
Privilege: Protected

Associations:

The DmGwV0BrowseRequest class has associations with the following classes:
None

7.3.29 DmGwV0ECSMapper Class

Parent Class: Not Applicable
Public: Yes
Distributed Object:Yes
Purpose and Description:
This class contains all the information and the operations required to map the V0 attributes to ECS and to map the ECS attributes to V0.

Attributes:

None

Operations:

DmGwV0ECSMapper - This operation constructs the Mapper class which performs the mapping from V0 to ECS and vice versa.
Arguments:
Return Type: Void
Privilege: Public

ECSToV0BrowseDescription - This operation maps the ECS browse image description to the V0 browse description format for a specific data collection.
Arguments: ECSBrowseDescription:Text
Return Type: String
Privilege: Public

ECSToV0BrowseType - This operation maps the ECS type of delivery for the browse image to the V0 type.
Arguments: ECSBrowseType:enum
Return Type: String
Privilege: Public

ECSToV0Campaign - This operation maps the ECS name of campaign or project that gathered the dataset to the V0 type.
Arguments: ECSFieldCampaign:String
Return Type: String
Privilege: Public

ECSToV0Contact - This operation maps the ECS point of contact reference for a data collection or browse image to the V0 format of contact reference.

Arguments:	ECSContactAddress:ContactStructure &
Return Type:	V0ContactStructure &
Privilege:	Public

ECSToV0DataCenter - This operation maps the acronym of the data center which transmits the data information to the V0 format.

Arguments:	ECSOrgName:String,ContactRole:String
Return Type:	String
Privilege:	Public

ECSToV0DataSetComment - This operation maps the ECS description for a particular data collection to the V0 format of comment information.

Arguments:	ECSDataCollectionDescription:Text
Return Type:	String
Privilege:	Public

ECSToV0DataSetRestriction - This operation maps the ECS ordering restriction and legal prerequisites placed on the data collection to the V0 restriction format.

Arguments:	ECSDataCollectionDescription:Text
Return Type:	String
Privilege:	Public

ECSToV0DatasetId - This operation maps the ECS name which identifies the data collection and associated granules to the V0 name.

Arguments:	ECSDataCollectionName:DataCollectionId &
Return Type:	String
Privilege:	Public

ECSToV0DateTime - This operation maps a particular ECS date and time format to the V0 data and time format.

Arguments:	ECSDatetime:DmGwDate &
Return Type:	DmGwDate &
Privilege:	Public

ECSToV0DayNight - This operation maps the ECS day/night indication to the V0 day/night indication format.

Arguments:	ECSDayNight:DayNight &
Return Type:	enum
Privilege:	Public

ECSToV0GranuleId - This operation translates the ECS universal reference to the individual granule in the data collection to the V0 format of granule id.

Arguments:	ECSGranuleId:GlUR &
Return Type:	String
Privilege:	Public

ECSToV0Parameter - This operation translates the ECS processing level to the V0 format of processing level.

Arguments:	ECSGeoPhysKeyword:String
Return Type:	GeoPhysParamList &
Privilege:	Public

ECSToV0ProcessingLevel - This operation maps the ECS sensor name to the V0 name.

Arguments:	ECSProcessingLevel:ProcessingLevel &
Return Type:	enum
Privilege:	Public

ECSToV0SensorName - This operation maps the ECS source name to the V0 name.

Arguments:	ECSInstrument:String
Return Type:	String
Privilege:	Public

ECSToV0SourceName - This operation translates the ECS spatial extent to the V0 format of spatial extent.

Arguments:	ECSSatellite:String
Return Type:	String
Privilege:	Public

ECSToV0SpatialExtent - This operation translates the ECS temporal extent to the V0 format of temporal extent.

Arguments:	ECSSpatial:DmGwSpatial &
Return Type:	DmGwV0Spatial &
Privilege:	Public

V0ToECSBrowseType - This operation maps the V0 name of campaign to the ECS type.

Arguments:	V0BrowseType:String
Return Type:	enum
Privilege:	Public

V0ToECSCampaign - This operation maps the V0 name which identifies the data collection and associated granules to the ECS name.

Arguments:	V0Campaign:String
Return Type:	String
Privilege:	Public

V0ToECSDatasetId - This operation maps a particular V0 date and time format to the ECS date and time format.

Arguments:	V0DateSet:String
Return Type:	DataCollectionId &
Privilege:	Public

V0ToECSDateTime - This operation maps the V0 day/night indication to the ECS day/ night indication format.

Arguments:	V0DateTime:DmGwDate &
Return Type:	DmGwDate &
Privilege:	Public

V0ToECSDayNight - This operation translates the V0 granule id to the ECS universial reference.

Arguments:	V0DayNight:enum
Return Type:	DayNight &
Privilege:	Public

V0ToECSGranuleId - This operation translates the V0 parameter to the ECS format of parameter.

Arguments:	V0GranuleId:String
Return Type:	GlUR &
Privilege:	Public

V0ToECSParameter - This operation translates the V0 processing level to the ECS format of processing level.

Arguments:	V0Parameter:String
Return Type:	GeoPhysParamList &
Privilege:	Public

V0ToECSProcessingLevel - This operation maps the V0 sensor name the the ECS name.

Arguments:	V0ProcessingLevel:enum
Return Type:	ProcessingLevel &
Privilege:	Public

V0ToECSSensorName - This operation maps the V0 source name to the ECS name.

Arguments:	V0SensorName:String
Return Type:	String
Privilege:	Public

V0ToECSSourceName - This operation translates the V0 spatial extent to the ECS format of spatial extent.

Arguments:	V0SourceName:String
Return Type:	String
Privilege:	Public

V0ToECSSpatialExtent - This operation translates the V0 temporal extent to the ECS format of temporal extent.
Arguments: V0Spatial:DmGwV0Spatial &
Return Type: DmGwSpatial &
Privilege: Public

V0ToECSTemporalExtent - This operation maps the V0 type of the browse image to the ECS type.
Arguments: ECSTemporal:DmGwTemporal &

V0ToECSTemporalExtent - This operation is the Mapper class destructor.
Arguments: V0Temporal:DmGwV0Temporal &
Return Type: DmGwTemporal &
Privilege: Public

~DmGwV0ECSMapper
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DmGwV0ECSMapper class has associations with the following classes:
None

7.3.30 DmGwV0IntBrowseRequest Class

Parent Class: DmGwV0ServRequest
Public: No
Distributed Object:No
Purpose and Description:
The DmGwV0IntBrowseRequest class is a specialization of the DmGwV0ServRequest class. This class is responsible for processing a V0 Browse request.

Attributes:

myAuthenticator - This attribute stores the authentication information.
Data Type: RWCString
Privilege: Private
Default Value:

myContactAddress - This attribute stores an address.
Data Type: DmGwAddress
Privilege: Private
Default Value:

myDatasetId - This attribute stores the name of the valid IMS dataset.
Data Type: RWCString
Privilege: Private
Default Value:

myGranuleId - This attribute stores the granule id.
Data Type: RWCString
Privilege: Private
Default Value:

myImageFileList - This attribute stores a list of file names which contained the returned images from data server.
Data Type: RWCString
Privilege: Private
Default Value:

myNumberImages - This attribute stores the number of returned images from data server.
Data Type: EcTInt
Privilege: Private
Default Value:

myODLFileList - This attribute stores a list of file names which contained the ODL tree returned from data server.
Data Type: RWCString
Privilege: Private
Default Value:

mySocketVec - This attribute stores a list of socket id.
Data Type: RWCString
Privilege: Private
Default Value:

myV0BrowseType - This attribute stores the type of delivery for browse image.
Data Type: RWCString
Privilege: Private
Default Value:

myV0DataCenterId - This attribute stores the data center id.

Data Type:	RWCString
Privilege:	Private
Default Value:	

Operations:

Abort - The Abort member function is a specialization of the DmGwV0ServRequest base class member function. It aborts any outstanding browse request and terminates further processing of this browse request.

Arguments:	Void
Return Type:	GlStatus
Privilege:	Public

BuildBrowseGranule - The BuildBrowseGranule member function inserts the granule data into myODLInputTree.

Arguments:	broRequestAggregate: AGGREGATE &, index: int
Return Type:	RWBoolean
Privilege:	Private

BuildContactAddress - The BuildContactAddress member function inserts the contact address into myODLInputTree.

Arguments:	broRequestAggregate: AGGREGATE &
Return Type:	RWBoolean
Privilege:	Private

BuildHeader - The BuildHeader member function inserts the header into myODLInputTree.

Arguments:	index: EcTInt
Return Type:	RWBoolean
Privilege:	Private

BuildODLInputTree - The BuildODLInputTree member function builds the myODLInputTree.

Arguments:	index: EcTInt
Return Type:	RWBoolean
Privilege:	Public

DmGwV0IntBrowseRequest - The DmGwV0IntBrowseRequest normal constructor.

Arguments:	
Return Type:	Void
Privilege:	Public

DmGwV0IntBrowseRequest - The DmGwV0IntBrowseRequest constructor to set all attributes.
Arguments: datacenterid: RWCString &, browsetype: RWCString &, datasetid: RWCString &, granuleid: RWCString &, authenticator: RWCString &, contactaddress: DmGwAddress &
Return Type: Void
Privilege: Public

ExtractImageInfo - The ExtractImageInfo member function extracts the image information (e.g. image size) from myODLOutputTree.
Arguments: ODLOutputree: AGGREGATE &, datasetid: RWCString &, granuleid: RWCString &, imageid: RWCString &, imagesize: RWCString &
Return Type: RWBoolean
Privilege: Public

ExtractResponse - The ExtractResponse member function extracts the images from returned ODL trees.
Arguments: imageList: GlParameterList &
Return Type: RWBoolean
Privilege: Public

GetCurrentGranule - The GetCurrentGranule member function returns the current processing granule id.
Arguments: index: EcTInt
Return Type: RWCString
Privilege: Private

GetCurrentODLFile - The GetCurrentODLFile member function returns the file name which contains the current processing ODL tree.
Arguments: index: EcTInt
Return Type: RWCString
Privilege: Private

GetNumberGranules - The GetNumberGranules member function returns the number of granule id the client requests.
Arguments: Void
Return Type: EcTInt
Privilege: Private

Status - The Status member function is a specialization of the DmGwV0ServRequest base class member function. It returns the current status of the browse request being processed.
Arguments: Void
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function is a specialization of the DmGwV0ServRequest base class member function. It transmits the request to V0 data server.

Arguments:	Void
Return Type:	GlStatus
Privilege:	Public

~DmGwV0IntBrowseRequest - The DmGwV0IntBrowseRequest destructor.

Arguments:	Void
Return Type:	Void
Privilege:	Public

Associations:

The DmGwV0IntBrowseRequest class has associations with the following classes:
None

7.3.31 DmGwV0InvRequest Class

Parent Class:	DmGwV0ServRequest
Public:	No
Distributed Object:	No

Purpose and Description:
The DmGwV0InvRequest class is a specialization of the DmGwV0ServRequest class. This class is responsible for processing a V0 inventory search.

Attributes:

myDatasetList - This attribute stores a list of the dataset ids for the inventory search.

Data Type:	RWCString
Privilege:	Private
Default Value:	

myGeoPhysParamList - This attribute stores a list of geophysical parameters for the inventory search.

Data Type:	RWCString
Privilege:	Private
Default Value:	

myGlobalGranuleOnly - This attribute stores the global granule only constraint for the inventory search.
Data Type: RWCString
Privilege: Private
Default Value:

myGranuleLimit - This attribute stores the granule limit constraint for the inventory search.
Data Type: RWCString
Privilege: Private
Default Value:

myProcessingLevel - This attribute stores the processing level constraint for the inventory search.
Data Type: RWCString
Privilege: Private
Default Value:

mySensorList - This attribute stores a list of sensor names for the inventory search.
Data Type: RWCString
Privilege: Private
Default Value:

mySourceList - This attribute stores a list of source names for the inventory search.
Data Type: RWCString
Privilege: Private
Default Value:

mySpatialExtent - This attribute stores the spatial constraint specified for the inventory search.
Data Type: DmGwSpatial *
Privilege: Private
Default Value:

mySpatialType - This attribute stores the spatial type constraint for the inventory search
Data Type: enum
Privilege: Private
Default Value:

myTemporalExtent - This attribute stores the temporal constraint specified for the inventory search.
Data Type: DmGwTemporal
Privilege: Private
Default Value:

myTemporalFlag - This attribute stores the flag to show if temporal constraint is specified.
Data Type: RWBoolean
Privilege: Private
Default Value:

Operations:

Abort - The Abort member function is a specialization of the DmGwV0ServRequest base class member function. It aborts any outstanding data server search request and terminates further processing of this inventory search.
Arguments: Void
Return Type: GlStatus
Privilege: Public

BuildHeader - The BuildHeader member function inserts the dataset, source, sensor, processing level, and geophysical parameter into myODLInputTree.
Arguments: Void
Return Type: RWBoolean
Privilege: Private

BuildODLInputTree - The BuildODLInputTree member function builds the ODL Tree based on the input constraints.
Arguments: Void
Return Type: RWBoolean
Privilege: Public

BuildSearchLocation - The BuildSearchLocation member function builds the Spatial extent in myODLInputTree.
Arguments: invSearchAggregate: AGGREGATE
Return Type: RWBoolean
Privilege: Private

DmGwV0InvRequest - The DmGWV0InvRequest normal constructor.
Arguments:
Return Type: Void
Privilege: Public

DmGwV0InvRequest
Arguments: DatasetList: RWCString, SourceList: RWCString, SensorList: RWCString, GeoPhyParamList: RWCString, ProcessingLevel: RWCString, GranuleLimit: EcTInt, TemporalExtent: DmGwTemporal, TemporalFlag: RWBoolean, SpatialExtent: DmGwSpatial *

ExtractDatasetInfo - The ExtractDatasetInfo member function extracts the dataset information from myODLOutputTree.
Arguments: datasetId: RWCString &, numberDatasets: int &
Return Type: RWBoolean
Privilege: Public

ExtractGranuleInfo - The ExtractGranuleInfo member function extracts the granule information from myODLOutputTree.
Arguments: granuleid: RWCString &, startdate: DmGwDateTime &, stopdate: DmGwDateTime &, browsetype: RWCString &
Return Type: RWBoolean
Privilege: Public

ExtractInvResults - The ExtractInvResults member function extracts the information from myODLOutputTree.
Arguments: InvResults: GlParameterList &
Return Type: RWBoolean
Privilege: Public

ExtractOutputHeader - The ExtractOutputHeader member function extracts the header from myODLOutputTree.
Arguments: Void
Return Type: RWBoolean
Privilege: Public

ExtractOutputLocation - The ExtractOutputLocation member function determines the type of spatial information contained in myODLOutputTree. Based on the spatial type, the appropriate extract member function will be invoked.
Arguments: Void
Return Type: RWBoolean
Privilege: Public

Status - The Status member function is a specialization of the DmGwV0ServRequest base class member function. It returns the current status of the inventory search being processed.
Arguments: Void
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function is a specialization of the DmGwV0ServRequest base class member function. It submits the request to V0 server.
Arguments: Void
Return Type: GlStatus
Privilege: Public

~DmGwV0InvRequest - The DmGwV0InvRequest destructor.
Arguments: Void
Return Type: Void
Privilege: Public

Associations:

The DmGwV0InvRequest class has associations with the following classes:
None

7.3.32 DmGwV0ProductRequest Class

Parent Class: DmGwV0ServRequest
Public: No
Distributed Object:No
Purpose and Description:
The DmGwV0ProductRequest class is a specialization of the DmGwV0ServRequest class. This class is responsible for processing a V0 Product request (i.e. data order request).

Attributes:

myBillingAddress - This attribute stores the user billing address.
Data Type: DmGwAddress
Privilege: Private
Default Value:

myCategory - This attribute stores the category constraint for the product request.
Data Type: RWCString
Privilege: Private
Default Value:

myContactAddress - This attribute stores the user contact address.
Data Type: DmGwAddress
Privilege: Private
Default Value:

myDAACContactAddress - This attribute stores the data center id constraint for the product request.
Data Type: DmGwAddress
Privilege: Private
Default Value:

myLineItemList - This attribute stores the granule order information (e.g. dataset id) for the product request.
Data Type: RWCString
Privilege: Private
Default Value:

myShippingAddress - This attribute stores the product shipping address.
Data Type: DmGwAddress
Privilege: Private
Default Value:

myType - This attribute stores the type constraint for the product request.
Data Type: RWCString
Privilege: Private
Default Value:

myV0DataCenterId - This attribute stores the data center id constraint for the product request.
Data Type: RWCString
Privilege: Private
Default Value:

myV0RequestId - This attribute stores the request id constraint for the product request.
Data Type: RWCString
Privilege: Private
Default Value:

Operations:

Abort - The Abort member function is a specialization of the DmGwV0ServRequest base class member function. It aborts any outstanding distribution request and terminates further processing of this product request.
Arguments: Void
Return Type: GlStatus
Privilege: Public

BuildAffiliation - The BuildAffiliation member function inserts the affiliation information into myODLInputTree.
Arguments: proRequestAggregate: AGGREGATE &
Return Type: RWBoolean
Privilege: Private

BuildContactAddress - The BuildContactAddress member function inserts the contact address into myODLInputTree.
Arguments: proRequestAggregate: AGGREGATE &
Return Type: RWBoolean
Privilege: Private

BuildHeader - The BuildHeader member function inserts the header information into myODLInputTree.
Arguments: Void
Return Type: RWBoolean
Privilege: Private

BuildLineItem - The BuildLineItem member function inserts the granule order data into myODLInputTree.
Arguments: proRequestAggregate: AGGREGATE &
Return Type: RWBoolean
Privilege: Private

BuildODLInputTree - The BuildODLInputTree member function builds the myODLInputTree.
Arguments: Void
Return Type: RWBoolean
Privilege: Public

DmGwV0ProductRequest - The DmGwV0ProductRequest normal constructor.
Arguments:
Return Type: Void
Privilege: Public

DmGwV0ProductRequest - The DmGwV0ProductRequest constructor to set all attributes.
Arguments: datacenterid: RWCString &, contactaddress: DmGwAddress &, category: RWCString &, type: RWCString &, lineitemlist: DmGwLineItem &
Return Type: Void
Privilege: Public

ExtractDAACContactAddress - The ExtractDAACContactAddress member function extracts the DAAC contact address from myODLOutputTree.

Arguments:	daaccontactaddress: DmGwAddress &
Return Type:	RWBoolean
Privilege:	Public

ExtractResponse - The ExtractResponse member function is responsible for building the V0 response message from myODLOutputTree.

Arguments:	Void
Return Type:	RWBoolean
Privilege:	Public

GetAddresses - The GetAddresses member function inquires the contact address, billing address, and shipping address from MSS.

Arguments:	Void
Return Type:	RWBoolean
Privilege:	Public

Status - The Status member function is a specialization of the DmGwV0ServRequest base class member function. It returns the current status of the product search being processed.

Arguments:	Void
Return Type:	GlStatus
Privilege:	Public

Submit - The Submit member function is a specialization of the DmGwV0ServRequest base class member function. It submits the request to V0 server.

Arguments:	Void
Return Type:	GlStatus
Privilege:	Public

~DmGwV0ProductRequest - The DmGwV0ProductRequest destructor.

Arguments:	Void
Return Type:	Void
Privilege:	Public

Associations:

The DmGwV0ProductRequest class has associations with the following classes:

None

7.3.33 DmGwV0Request Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The DmGwV0Request object is the base class from which specialized versions (e.g. DmGwV0BrowseRequest) are derived. DmGwV0Request abstracts the operations and attributes that are common to all gateway V0 request processing. DmGwV0Request objects are created by the V0ServerFrontEnd component.

Attributes:

myODLInputTree - stores the internal representation of the ODL message received from the V0 Client.
Data Type: ODLTree*
Privilege: Protected
Default Value:NULL

myODLOutputTree - stores the internal representation of the ODL response message to be delivered to the V0 Client.
Data Type: ODLTree*
Privilege: Protected
Default Value:NULL

myV0MesageId - stores the V0 Message Id from myODLInputTree.
Data Type: String
Privilege: Protected
Default Value:

myV0Monitor - stores the V0 Monitor information from myODLInputTree.
Data Type: ODLAggregate
Privilege: Protected
Default Value:

myV0Version - stores the V0 Version information from myODLInputTree.
Data Type: ODLAggregate
Privilege: Protected
Default Value:

Operations:

Abort - The Abort member function stops all further processing of the DmGwV0Request object.
Arguments:
Return Type: GlStatus
Privilege: Public

DmGwV0Request - The DmGwV0Request constructor initializes the myODLOutputTree attribute from the inputODLTree argument and extracts the common V0 ODL attributes. Upon completion, the DmGwV0Request object is ready to begin processing the request.
Arguments: inputODLTree:ODLAggregate
Return Type: Void
Privilege: Public

Status - The Status member function returns the current status of the DmGwV0Request object via a GlStatus object.
Arguments:
Return Type: GlStatus
Privilege: Public

Submit - The Submit member function initiates processing of the DmGwV0Request class. After the object has been constructed, the Submit operation begins the actual request processing.
Arguments:
Return Type: GlStatus
Privilege: Public
This is an abstract operation

V0RequestComplete - The V0RequestComplete member function schedules the object for garbage collection.
Arguments:
Return Type: void
Privilege: Public

V0RequestFailed - The V0RequestFailed member function stops further processing, and schedules the DmGwV0Request object for garbage collection.
Arguments: whatHappened:GlStatus
Return Type: void
Privilege: Public

acknowledgeChunk - The acknowledgeChunk member function is called by the V0ServerFrontEnd code whenever a previously delivered chunk is acknowledged by the V0 Client.

Arguments:	
Return Type:	Bool
Privilege:	Protected

extractV0MessageId - The extractV0MessageId initializes the myV0MessageId attribute from information contained in myODLInputTree.

Arguments:	
Return Type:	Bool
Privilege:	Protected

extractV0Monitor - The extractV0Monitor member function extracts the V0 Monitor information from myODLInputTree and initializes the myV0Monitor attribute.

Arguments:	
Return Type:	Bool
Privilege:	Protected

extractVersion - The extractVersion member function initializes the myV0Version attribute from information contained in myODLInputTree.

Arguments:	
Return Type:	Bool
Privilege:	Protected

Associations:

The DmGwV0Request class has associations with the following classes:

Class: DmGwRequestList
Class: V0ServerFrontEnd initieates - The V0ServerFrontEnd initiates a DmGwV0Request.
Class: V0ServerBackEnd returnsresultsusing - The DmGwV0Request returns results to the V0 client using the V0ServerBackEnd.

7.3.34 DmGwV0Requests Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class keeps track of the V0 requests coming in and their status to perform housekeeping.

Attributes:

MessageId - Vo client generated message id.
Data Type: Char(30)
Privilege: Public
Default Value:

Status - Current status of the message.
Data Type: int
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwV0Requests class has associations with the following classes:
None

7.3.35 DmGwV0ServRequest Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The DmGwV0ServRequest object is the base class from which specialized versions (e.g. DmGwV0IMSBrowseRequest) are derived. DmGwV0ServRequest abstracts the operations and attributes that are common to all gateway V0 request processing.

Attributes:

myODLInputTree - This attribute stores the internal representation of the ODL message received from the V1 Client (to be delivered to V0 Data Server).
Data Type: AGGREGATE
Privilege: Protected
Default Value:NULL

myODLOutputTree - This attribute stores the internal representation of the ODL response message delivered from the V0 Data Server.
Data Type: AGGREGATE
Privilege: Protected
Default Value:NULL

myStatus - This attribute stores the current status of the DmGwV0ServRequest object.
Data Type: GlStatus
Privilege: Protected
Default Value:

myV0MessageId - This attribute stores the V0 message id constraint.
Data Type: RWCString
Privilege: Protected
Default Value:

Operations:

Abort - The Abort member function stops all further processing of DmGwV0ServRequest object.
Arguments: Void
Return Type: GlStatus
Privilege: Public

CreateMessageId - The CreateMessageId member function inserts the V0 Message Id into myODLInputTree.

Arguments:	Void
Return Type:	RWBoolean
Privilege:	Public

DmGwV0ServRequest - The DmGwV0ServRequest normal constructor.

Arguments:	
Return Type:	Void
Privilege:	Public

InsertMonitor - The InsertMonitor member function inserts the V0 Monitor information into myODLInputTree.

Arguments:	Void
Return Type:	RMBoolean
Privilege:	Public

InsertVersion - The InsertVersion member function inserts the V0 Version information into myODLInputTree.

Arguments:	myAggregate: DmGwODLAggregate
Return Type:	RWBoolean
Privilege:	Public

Status - The Status member function returns the current status of the DmGwV0ServRequest object.

Arguments:	Void
Return Type:	GlStatus
Privilege:	Public

~DmGwV0ServRequest - The DmGwV0ServRequest destructor.

Arguments:	Void
Return Type:	Void
Privilege:	Public

Associations:

The DmGwV0ServRequest class has associations with the following classes:

Class: DmImV0ServIF
Class: V0Server

7.3.36 DmGwV0StatusMessage Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class stores the list of V0 status codes and the associated messages.

Attributes:

StatusCode - V0 status code.
Data Type: int
Privilege: Public
Default Value:

StatusMessage - Descriptive message what the status code means.
Data Type: Char(255)
Privilege: Public
Default Value:

Operations:

None

Associations:

The DmGwV0StatusMessage class has associations with the following classes:
Class: DmGwStatusCodeMap usestofinddescriptionofthev0statuscode

7.3.37 DmImV0ServIF Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:

Attributes:

None

Operations:

None

Associations:

The DmImV0ServIF class has associations with the following classes:
Class: DmGwV0ServRequest

7.3.38 DsClDescriptor Class

Parent Class:	Not Applicable
Public:	Yes
Distributed Object:	Yes

Purpose and Description:
The DsClDescriptor class is the proxy class which is imported from the Data Server Subsystem. The description of the server class is defined in the DID305 Data Server Subsystem section.

Attributes:

None

Operations:

None

Associations:

The DsClDescriptor class has associations with the following classes:
None

7.3.39 DsClESDTReference Class

Parent Class: Not Applicable
Public: Yes
Distributed Object: Yes
Purpose and Description:
The DsClESDTReference class is the proxy class which is imported from the Data Server Subsystem. The description of the server class is defined in the DID305 Data Server Subsystem section.

Attributes:

None

Operations:

None

Associations:

The DsClESDTReference class has associations with the following classes:
None

7.3.40 DsClESDTReferenceCollector Class

Parent Class: Not Applicable
Public: Yes
Distributed Object: Yes
Purpose and Description:
The DsClESDTReferenceCollector class is the proxy class which is imported from the Data Server Subsystem. The description of the server class is defined in the DID305 Data Server Subsystem section.

Attributes:

None

Operations:

None

Associations:

The DsClESDTReferenceCollector class has associations with the following classes:
None

7.3.41 DsClQuery Class

Parent Class: Not Applicable
Public: Yes
Distributed Object: Yes
Purpose and Description:
The DsClQuery class is the proxy class which is imported from the Data Server Subsystem. The description of the server class is defined in the DID305 Data Server Subsystem section.

Attributes:

None

Operations:

None

Associations:

The DsClQuery class has associations with the following classes:
None

7.3.42 DsClRequest Class

Parent Class: Not Applicable
Public: Yes
Distributed Object:Yes
Purpose and Description:
The DsClESDTRequest class is the proxy class which is imported from the Data Server Subsystem. The description of the server class is defined in the DID305 Data Server Subsystem section.

Attributes:

None

Operations:

None

Associations:

The DsClRequest class has associations with the following classes:
None

7.3.43 V0Server Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:

Attributes:

None

Operations:

None

Associations:

The V0Server class has associations with the following classes:
Class: DmGwV0ServRequest

7.3.44 V0ServerBackEnd Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The V0ServerBackEnd is an OTS package provided by the V0 IMS Server.

Attributes:

None

Operations:

None

Associations:

The V0ServerBackEnd class has associations with the following classes:
Class: DmGwV0Request returnsresultsusing - The DmGwV0Request returns results to the V0 client using the V0ServerBackEnd.

7.3.45 V0ServerFrontEnd Class

Parent Class: Not Applicable
Public: No
Distributed Object:No
Purpose and Description:
The V0ServerFrontEnd is an OTS package provided by the V0 IMS Server.

Attributes:

None

Operations:

None

Associations:

The V0ServerFrontEnd class has associations with the following classes:

Class: DmGwRequestList addstp - The V0ServerFrontEnd adds requests to the DmGwRequestList.

Class: DmGwV0Request initieates - The V0ServerFrontEnd initiates a DmGwV0Request

7.4 CSCI Structure

Table 7.4-1 provides a summary of the components which make up this CSCI. Since this CSCI is on an incremental development track, the table presents the current best estimate of the CSCI components. These components are likely to change as the CSCI evolves.

Table 7.4-1. Gateway CSCI Components

Name	Description	Type (DEV or OTS)
Gateway server	Application server that processes the gateway requests	DEV
V0 Front End	System level V0 IMS server libraries for receiving messages.	OTS
V0 Back End	V0 System level IMS libraries for sending messages.	OTS
V0 External Interface	The set of classes that interface to the V0 IMS libraries to call the V0 and receive messages from the V0 server.	DEV
Configuration and Setup.	The GUI used to configure and set up the V0 Gateway.	DEV

7.5 CSCI Management and Operation

The Data Dictionary Maintenance Tool is used by operators to configure the scope of the GTWAYS's data and service access. It is used to ingest new Valids files from V0 and incorporate those in the DDICT. The operator can then map the V0 terms and attributes to the ECS terms and attributes.

The GTWAY Configuration and Setup Tool is used to set the options for the GTWAY. The GTWAY Configuration and Setup Tool can be used to manually force a reread of the DDICT service. This would be used, when the operator wanted the changes to the DDICT service to take effect in the GTWAY. Another function of the GTWAY Configuration and Setup Tool is to define run-time parameters such as maximum number of sessions, result set memory cache per session, etc.

8. DMGHW - Data Management HWCI

The Data Management HWCI (DMGHW) is the primary HWCI within CIDM. The DMGHW CI provides the necessary hardware resources for the persistent storage of dictionary and schema data. The DMGHW CI will be sized to support the service demands of the Advertising CI and Gateway CI services in Release A, and will expand to include the Data Dictionary CI, Local Information Manager CI and Distributed Information Manager CI services in Release B. The primary technologies that apply to the DMGHW CI include DBMS servers, World Wide Web servers, host attached disk, possible use of RAID disk and a variety of communications capabilities. A small pool of local operations workstations will support DBMS management, data repository administration, data specialist, user support and phone/mail support functions. The operations workstations are few in number and small in scale. The scope of DBMS management, data repository administration, data specialist, user support and phone/mail support functionality will be explored further as the physical database analysis matures in the future.

8.1 HW Design Drivers

The design of the DMGHW CI is primarily based on the volume of service "pull" that will be generated by the user community and the processing load that is placed on the Advertising Service CI, Gateway CI, Data Dictionary CI, Local Information Manager CI and Distributed Information Manager CI DBMSs as a result. The following is a list of the major contributing sources that drive the design of the DMGHW CI:

- User community access profiles as predicted by User Modeling analysis
- Data Modeling and Data Architecture analysis depicting what classes of information are held, and within what system components
- COTS software selection of DBMS and HTTP servers
- Core system functional and performance requirements

In short, the DMGHW CI must support user demand for: 1) V0 data access, 2) World Wide Web services 3) DBMS services. Design recommendations provided at this time are geared toward Release B.

8.1.1 Key Trade-off Studies and Prototypes

The following Trade Studies and/or Technical Papers were used as follows:

1) User Pull Analysis Notebook. Defined rate (number per minute) of user search requests across all DAACs (60-TP-004-001).
2) Distributed Database Architecture of Data Management Subsystem - Technical Paper. Defined approach for implementing a distributed database architecture across multiple DAACs (430-TP-004-001).
3) DBMS Benchmark Report - Technical Paper. Provides performance evaluation of DBMS COTS packages (430-TP-003-001).

8.1.1.1 Prototype Studies

The following prototypes are pertinent to the Release A Data Management HWCI design:

- Advertising Service CI Prototype: At this time limited performance evaluations have been performed on the Advertising CI prototype; however, one of the goals for the Advertising CI EP6 prototype will be to obtain realistic performance characteristics and use them as inputs to the Hardware Design. These same inputs will be used as a model for Data Dictionary operations.
- Gateway CI Prototype: The first Gateway CI prototype was functional in the Release A CDR time frame; therefore, one of the goals (post Release A CDR) will be to obtain realistic performance characteristics and use them as inputs to the Hardware Design.
- LIM Prototype: The LIM prototype will be functional during PW2. After the workshop, the LIM prototype will be used to obtain performance characteristics for LIM and DIM operations. These initial performance characteristics will not include coincident search operations. Coincident search operation characteristics will be captured after EP7 when the coincident search function is implemented.

8.1.2 Sizing and Performance Analysis

At this time, preliminary performance analysis data is being used to calculate the size of the DMGHW CI DBMS/Web server. The performance data is derived from User Modeling, Vendor, and DBMS Technical Paper inputs. As the Incremental Track software design matures and is subjected to future prototyping activities, performance results will be revised accordingly. The operations support workstations are being sized according to operations staffing and functionality requirements for each individual DAAC site.

Processor Capacity Sizing:

Processor sizing for the DMGHW CI is dependent on Incremental Track Development prototyping, vendor inputs for processor ratings in Transactions Per Minute (TPM), and User Modeling data for projected rate of search requests (transactions) at each DAAC site. A performance profile of the transactions to be performed as a function of the Advertising Service CI and Data Dictionary is to be determined as part of the EP6 and EP7 incremental development. A performance profile of the transactions to be performed as a function of the LIM CI is to be determined as part of the PW2 prototype. The profile will be extended at EP7 to include coincident search operations. The LIM CI characteristics will be used as a model for the DIM CI profile.

Vendor performance inputs are available in the Data Management section of the DAAC-specific volumes, for the Release B sites that are configured with operational DM functionality (LaRC, GSFC, EDC, ORNL, JPL, ASF, NSIDC).

User Modeling analyses that provide sizing input to the DMGHW CI include number of concurrent users and/or concurrent service requests per DAAC per time frame, and frequency of searches per DAAC per time frame. Peak and nominal service request arrival rates are also provided by the User Modeling source data. The DMGHW CI processor capacity sizing has been achieved by deriving the number of transactions per second (in relation to the CPU) to be performed by each individual search request and multiplying out by the frequency with which the search requests are invoked. User Modeling search classes/types that will be invoked by the user community have been defined by the User Characterization Group. The following is a list of search types that will be supported by the Advertising Service CI, Gateway CI, Data Dictionary, Local Information Manager CI and Distributed Information Manager CI services (as defined by the User Characterization Team):

- Simple search /1
- Simple search /M
- Match-up search /1
- Match-up search /M
- Coincident search /1
- Coincident search /M

The User Characterization analysis results are available in the Data Management section of the DAAC specific volumes.

8.1.2.1 HWCI Alternatives

The following processor classes are been evaluated for Release B:

- Low-End Symmetric Multiprocessor (SMP)–Low-End SMP servers provide excellent scalability (from either 1-2, or 1-4 CPUs), graceful performance degradation, load balancing and excellent cost/performance benefits. SMPs offer higher internal bus bandwidths and are capable of running DBMS processes in parallel across multiple processors.
- Uniprocessor Server–Uniprocessor workstations are low cost, single CPU designs. Uniprocessor workstations allow for small concurrent user loads, and offer very little in terms of scalability. The Uniprocessor workstations will be used for DBMS management, data repository administration, data specialist, user support and phone/mail support functions.

Disk storage capacity sizing for the DMGHW CI was determined for each DAAC site based on preliminary DBMS sizing efforts for the Advertising Service CI, Gateway CI, Data Dictionary CI, Local Information Manager CI and Distributed Information Manager CI operational databases plus vendor inputs for the following COTS software: 1) DBMS software, 2) HTTP server software, 3) Operating System software, 4) Communications and Utilities software. Estimates for total disk storage capacity is available in the DAAC specific volumes.

8.1.3 Scalability, Evolvability and Migration to Release B

The DMGHW CI is designed with sufficient scalability in order to make the transition from Release A to Release B a smooth one. The DMGHW CI will support three additional Application CIs in Release B: 1) Data Dictionary CI, 2) Local Information Manager CI, 3) Distributed Information Manager CI. A significant increase in the volume of user traffic will also impact the DMGHW CI in Release B.

Processing:

In terms of processing scalability, a low-end SMP server clearly out performs a uniprocessor server architecture. The uniprocessor architecture is not very flexible and does not offer much in terms of future processing expansion, where as the low-end SMP is very flexible since multiple processors can be added. Since the SMP can be configured with additional processors it offers significant future growth capability and investment protection, because the cost of adding an additional CPU is much less than purchasing an additional server. The SMP architecture is also much more likely to be supported by technology advancements (technology refreshes) that include processor upgrades. For example, companies such as Hewlett Packard offer well defined upgrade paths for their SMP architecture's that include processor upgrades to future chip designs, but many of HP's uniprocessor systems will not be supported by future processor upgrades.

Since DBMS software servers can be run in parallel over multiple processors, it makes much more sense to implement an SMP architecture when a DBMS is the definitive process in the design, such as in the case of the DMGHW CI. Running a DBMS software server in parallel across multiple processors is dependent on vendor specific software implementations, and is not an automatic function of a DBMS server design.

Implementing a low-end SMP server architecture in Release A is more than adequate for the Release A DAAC sites; however, given that the volume of hits on the Advertising Service CI, LIMGR CI and DIMGR CI DBMSs (in Release B) is largely unknown at this point, an SMP provides a resource buffer for immediate expansion in processing. The low-end SMP architecture also provides good scalability in relation to the addition of three software CIs in Release B: 1) Data Dictionary CI, 2) Local Information Manager CI, 3) Distributed Information Manager CI. Furthermore, a low-end SMP architecture satisfies the design requirement of 100% growth in capacity with minimal impact to the physical components of the DMGHW CI server. The only way to satisfy this requirement with a uniprocessor architecture is to purchase an additional server, which is not cost effective. Implementing a low-end SMP architecture in Release A makes the migration to Release B simplistic since configuration changes for Release B would build on the Release A design and; therefore, be minimal in scope. The impact on the Operations Staff at each DAAC facility will also be minimal since the base configuration would not change. Given that the amount of time between Release A and Release B missions/operations is relatively short, the DMGHW CI server hardware design has been sized according to Release B design goals.

The DMGHW CI sizing rationale and configuration plan is available in the DAAC specific volumes.

Storage:

The current DMGHW CI design allows for both standard and RAID disk application without re-work of the core design. This can be accomplished through the application of channel adapted disk capacity upgrades (e.g., larger RAID units), or through network adapted disk servers addressable by the DBMS servers. These growth decisions, which are likely to occur at the DAAC due to science mission growth and/or expansion, will be made through predictions on future DBMS transaction rate needs, as well as I/O and storage requirements.

8.2 HWCI Structure

The DMGHW CI consists of three major components that support DBMS processing, DBMS management, data specialists and user support functions. A block diagram for the Release B DMGHW CI configuration is shown in Figure 8.2-1. The block diagram illustrates the physical layout of the DMGHW CI and the interconnection between different components.

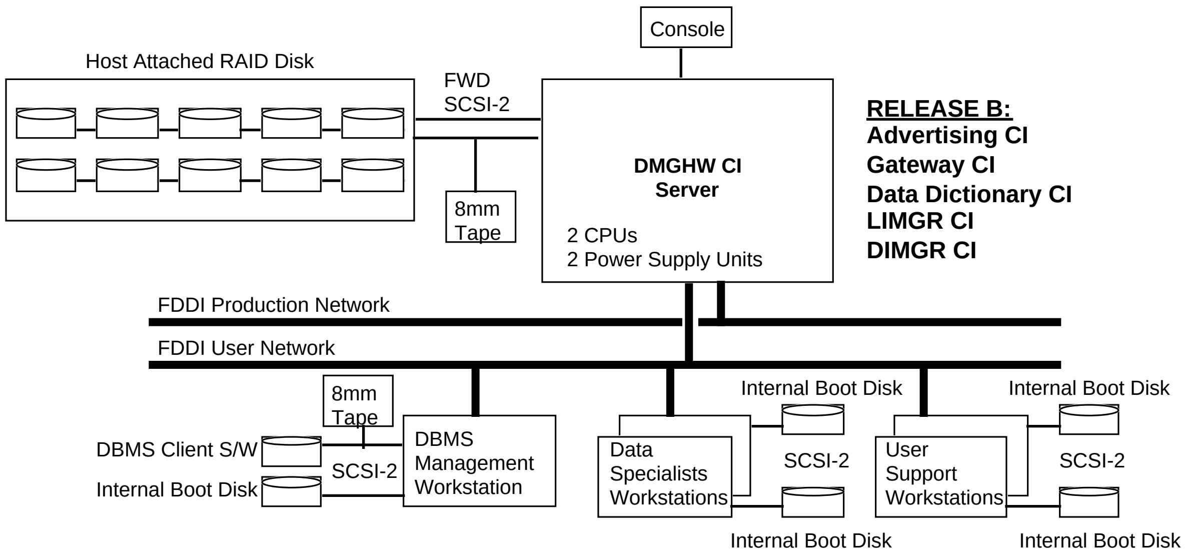


Figure 8.2-1. Data Management HWCI

8.2.1 Connectivity

Local Disk Connectivity:

The DMGHW CI servers and workstations will be utilizing host attached internal disks and a host attached RAID disk unit that are connected via Fast/Wide SCSI-2 in Release B.

Network Connectivity:

The DMGHW CI servers and workstations will be directly connected to the local DAAC FDDI network. The DMGHW CI hosts will be connected to the same FDDI ring as the Data Server hosts, as is illustrated in the Data Management Network Connectivity Diagram (Figure 8.2.1-1). Each DMGHW CI host will contain dual-attached station (DAS) cards, which will be dual-homed to separate FDDI concentrators. This provides redundancy so that full connectivity will exist to the DMGHW CI servers in the event of a concentrator failure. The workstations will contain single-attached station (SAS) cards and each will be connected to a single concentrator, but they will be split across the FDDI concentrators so that they are not all connected to the same concentrator. The FDDI concentrators are, in turn, connected to the local DAAC FDDI switch (Refer to Section 8.2 of Volume 0 for a general description of DAAC networks). A diagram depicting how the DMGHW CI will be connected to the local DAAC FDDI network is shown in the following network connectivity diagram (Figure 8.2.1-1).

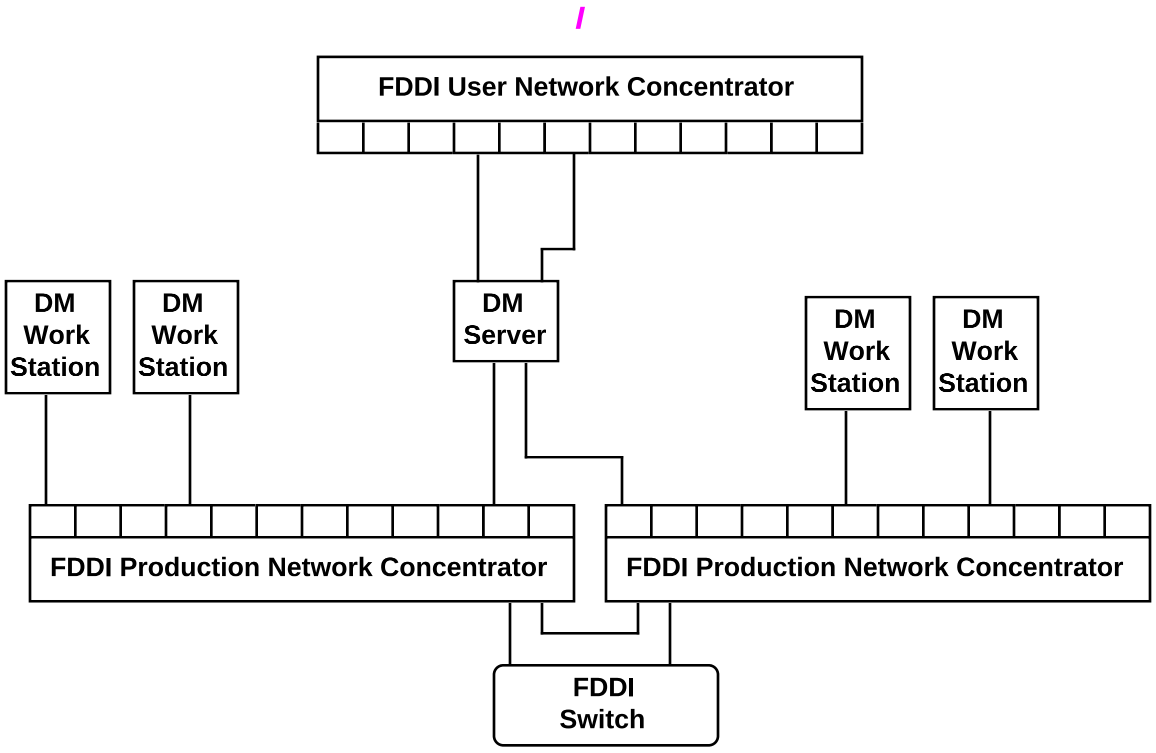

FDDI User Network Concentrator
DM Work Station
DM Work Station
DM Server
DM Work Station
DM Work Station
FDDI Production Network Concentrator
FDDI Production Network Concentrator
FDDI Switch

Figure 8.2.1-1. Data Management Network Connectivity

8.2.2 HWCI Components

The hardware associated with the DMGHW CI consists of a low-end SMP server, low-end uniprocessor workstations, RAID disk, and 8mm tape drives used in support of Release B Advertising Service CI, Gateway CI, Data Dictionary CI, Local Information Manager CI and Distributed Information Manager CI configurations. The number of physical components, and whether or not certain components will be used, is dependent on each DAAC specific configuration. Table 8.2.2-1 provides an overview of the major physical components of the DMGHW CI.

8.3 Failover and Recovery Strategy

The DMGHW CI server will meet Reliability, Maintainability and Availability requirements as stated in "Availability Models/Predictions for the ECS Project" (515-CD-001-003), "Maintainability Predictions for the ECS Project" (518-CD-001-003), and "Reliability Predictions for the ECS Project" (516-CD-001-003) documentation. The DMGHW CI will be designed to meet the following RMA requirements for Release A and Release B:

(1) Availability: 0.993

(2) Mean Down Time: <2 Hrs.

Table 8.2.2-1. Data Management HWCI Components

Component Name	Class/Type	Comments
DBMS/Web Server (Processing)	Low-End SMP Server (1-4 CPUs)	Designed to be versatile with regard to scalability, and meet moderate to high processing loads. Used at sites with moderate to heavy volume of user (search) traffic. Would provide processing, I/O and disk resources in support of the Advertising Service CI, GTWAY CI, DDCICT CI, LIMGR CI and DIMGR CI in Release B.
	Host Attached RAID Disk	RAID disk used for storage in support of DBMS software configuration. Designed for high availability, but not necessarily required for Release A due to the static nature of the Advertising and Gateway CI databases.
	8mm Tape Drive	8mm tape drive used to backup databases and perform other routine maintenance functions. One 5-7 GB unit connected to each DBMS server host.
DBMS Management Workstation	Low-End Uniprocessor Workstation	Designed to meet small processing loads. Used to support management activities to be performed on databases by the DataBase Administrator (DBA). Will provide processing, I/O and disk resources to the DBA.
	8mm Tape Drive	8mm tape drive used to perform routine DBA maintenance functions (one 5-7 GB unit).
Data Specialist /User Support Workstations	Low-End Uniprocessor Workstations	Used in support of Data Specialist and User Support functions.

8.3.1 Server Hardware Failure Recovery

The DMGHW CI will contain two DBMS servers in Release A. One server will be the primary, or "active" server and the other will be the secondary, or "standby" server. In the event of a hardware failure on the active server, the standby server will be configured so that it can readily assume the processing functions, or role of the active server so as to meet availability requirements. The failure recovery strategy for Release A; therefore, is simply to activate the standby server in the event of a failure to the active server. For Release B a single host will satisfy the processing requirements of the IOS and DM CIs. The Release B single server configuration will offer redundancy in the form of: 1) dual processors, 2) dual power supply units, 3) dual FWD SCSI-2 cards, and 4) dual FDDI network cards 5) duplicate OS boot/application disk. The components listed above are hot swapable units which allow them to be replaced without shutting down the server. Also, the HP-UX operating system features memory page de-allocation which automatically blocks out any portion of memory in which an error has been detected; therefore, a failure to memory will not bring operations to a halt. The single server host configuration will allow applications to be run in parallel across both processors, which enhances load balancing and availability/recovery capabilities. The DMGHW CI DBMS/Web server will automatically reconfigure itself in a single CPU configuration in the event of failure to a single CPU. The dual power supply units, FWD SCSI-2 cards and FDDI network cards will also provide continued availability in the case of failure to a single component (per function). The redundant configuration of the DMGHW CI DBMS/ Web server has been analyzed by the ECS Reliability Engineering Group using COTS vendor provided data to ensure that all functional availability requirements are met. Preliminary analysis results revealed that the DMGHW CI DBMS/Web server has an MTBF (mean-time-between-failures) of greater than 20,000 hours which meets all pertinent RMA requirements. The DMGHW CI will automatically failover / recover due to its redundant configuration. The automatic failover / recovery design will protect mission critical applications from a wide variety of hardware and software failures. For example, the following failures are readily detected and responded to: 1) System processors, 2) System memory, 3) LAN interfaces, 4) FWD SCSI-2 interfaces 5) System processes, 6) Application processes. The design that has been described includes no single point of failure (excluding the system backplane) assuming that data disk drives are mirrored and multiple LAN interconnects are used (for example dual homed FDDI). Failure and recovery configurations for the DMGHW CI are discussed further in the DAAC specific volumes.

8.3.2 DBMS Failure Recovery

The DMGHW CI will house Advertising Service CI, Gateway CI, Data Dictionary CI, Local Information Manager CI and Distributed Information Manager CI software in Release B. DMGHW CI DBMS recovery techniques are currently being analyzed. DBMS Replication Server strategies, along with mirrored disk strategies, are being scrutinized and will continue to be explored further as the physical database analysis matures in the future. Databases will be backed up onto tape media on a regular basis to ensure a complete recovery capability.

Disk Mirroring:

Disk mirroring provides a form of redundancy to protect against hardware failure and to provide a degree of fault tolerance. Disk mirroring is the capability to maintain a replicate of all data stored on a disk media device. Disk mirroring can provide non-stop recovery in the event of a disk failure. With respect to disk failure recovery, the DMGHW CI server will have multiple physical disks mirrored in order to ensure safe storage of production DBMSs. The disks will be grouped into two logical sets and the second set will mirror the first set. There are two basic strategies that can be employed:

(1) *Hardware supported*–physical cross strapping of disk between two servers, with one acting as the primary disk set and the second acting as the backup, or mirror set.

(2) *Primarily software based*–disk mirroring between primary and secondary sets of disks through the use of disk mirroring protocols.

Processes such as Sybase replication server or Sybase disk mirroring capability can also be executed to provide disk mirroring capability. A Sybase SQL server database device can be duplicated, i.e. all writes to the device are copied to a separate physical device so that if one of the devices fail, the other contains an up-to date copy of all transactions. Sybase replication server and disk mirroring capability is currently under evaluation to determine if a method of deployment is necessary given RAID disk alternatives.

When deciding to mirror disks, one must weigh factors such as the cost of system downtime, possible degradation of performance, and the cost of storage media. Considering these issues, one has to decide what devices to mirror; selected devices such as transaction logs, or all devices. Mirroring selected devices minimizes disk resources and performance degradation but requires manual intervention to restore the un-mirrored devices from backup in the event of a disk hardware failure. Mirroring all DBMS disk devices such as the master device, user databases, and transaction logs provides a non-stop recovery from hardware failure, but provides a slight degradation in performance. In the event of disk media failure, the mirror device can take over, typically without any downtime. When the damaged device is repaired or replaced, it can then be re-synchronized with the other undamaged, operational devices.

The recommended DMGHW CI disk mirroring configuration is documented in the DAAC specific volumes.

8.3.3 Network Recovery

There are three types of network failures that may affect the DMGHW CI: 1) A FDDI cable failure due to physical damage would require a new cable to be installed, 2) If an individual port on the FDDI concentrator fails, then the attached host would have to be routed to another port, 3) If the entire FDDI concentrator fails, then it would have to be replaced (which can be done rapidly since the units require very little configuration).

Note that the failures described above result in service interruption only to the workstations. Given that the DBMS server hosts will be attached to separate concentrators via a dual-attached station (DAS) and single attached station (SAS) card, they will communicate as normal in the event of a single cable, or single concentrator fault; therefore, critical applications will not be affected by the failure.

This page intentionally left blank.

Abbreviations and Acronyms

ACMHW	Access Control and Management HWCI
ADC	Affiliated Data Center
ADS	Archive data sets
ADSHW	Advertising Service HWCI
ADSRV	Advertising Service CSCI
AITHW	Algorithm Integration & Test HWCI
AITTL	Algorithm Integration and Test Tools (CSCI)
AM	Ante meridian
ANSI	American National Standards Institute
APC	Access/Process Coordinators
API	Application Programming Interface
APID	Application Process Identifier
AQAHW	Algorithm QA HWCI
ASAP	As soon as possible
ASCII	American Standard Code for Information Interchange
ASF	Alaska SAR Facility (DAAC)
ATM	Asynchronous Transfer Mode
CD ROM	Compact disk read only memory
CDRL	Contract Data Requirements List
CERES	Clouds and Earth's Radiant Energy System
CI	Configuration Item
CIESIN	Consortium for International Earth Science Information Network
CLS	Client Subsystem
COTS	Commercial off-the-shelf
CPU	Central processing unit
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CCSDS	Consultative Committee for Space Data Systems
CM	Configuration Management
CSDT	Computer Science Data Types
CSMS	Communications and Systems Management Segment
CSS	Communication Subsystem (CSMS)

DAA	DAN Acknowledge
DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DAN	Data Availability Notice
DAO	Data Assimilation Office
DAR	Data Acquisition Request
DAS	Data Availability Schedule
DBA	Database administrator
DBMS	Database Management System
DDA	Data Delivery Acknowledgment
DDICT	Data Dictionary CSCI
DDIST	Data Distribution CSCI
DDN	Data Delivery Notice
DDSRV	Document Data Server CSCI
DESKT	Desktop CI
DEV	Developed code
DID	Data Item Description
DIM	Distributed Information Manager
DIMGR	Distributed Information Management CSCI
DIPHW	Distribution & Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management System
DMS	Data Management Subsystem
DP	Data Processing
DPR	December Progress Review
DPREP	Science Data Pre-Processing CSCI
DPS	Data Processing Subsystem
DR	Data Repository
DRPHW	Data Repository HWCI
DS	Data Server
DSM	Distribution Storage Management
DSS	Data Server Subsystem
DT	Data Type
ECS	EOSDIS Core System
EDC	EROS Data Center (DAAC)
EDOS	EOS Data and Operations System

EOC	Earth Observation Center (Japan)
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
EP	Evaluation Package
EP	Early Prototype
ESDIS	Earth Science Data and Information System
ESDT	Earth Science Data Types
F&PRS	Functional and Performance Requirements Specification
FC	Fiber Channel
FDDI	Fiber distributed data interface
FDF	Flight Dynamics Facility
FOS	Flight Operations Segment
FSMS	File and Storage Management System
Ftp	File transfer protocol
GB	Gigabyte
GDAO	GSFC Data Assimilation Office
GFLOPS	Giga (billions) Floating Point Operations per Second
GOES	Geostationary Operational Environmental Satellite
GRIB	Gridded Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	Graphic user interface
HDF	Hierarchical Data Format
HiPPI	High Performance Parallel Interface
HMI	Human machine interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HWCI	Hardware Configuration Item
I&T	Integration and Test
I/O	Input/Output
ICD	Interface Control Document
ICLHW	Ingest Client HWCI
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IERS	International Earth Rotation Service
IMS	Information Management Subsystem

IP	International Partner
IR-1	Interim Release 1
IRD	Interface Requirements Document
IS	Ingest Subsystem
ISS	Internetworking Subsystem (CSMS)
JPL	Jet Propulsion Laboratories
LaRC	Langley Research Center
LIM	Local Information Manager
LIMGR	Local Information Management CSCI
LIS	Lightning Imaging Sensor
L0	Level 0
MB	Megabyte
Mbps	Megabits per second
MBps	Megabytes per second
MD	Maryland
MFLOP	Millions of Floating Point Operations per Second
MOC	Mission Operations Center
MODIS	Moderate-Resolution Imaging Spectrometer
MPP	Massively Parallel Processor
MRF	Medium Range Forecast
MSFC	Marshall Space Flight Center
MSS	Management Subsystem (CSMS)
MTBF	Mean time between failures
MTTR	Mean time to restore
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	Orbit/Attitude
ODC	Other Data Center
ODL	Object Description Language
ORNL	Oak Ridge National Laboratory (DAAC)
OSM	Open Storage Manager
OTS	Off-the-shelf
PAM	Permanent Archive Manager
PCI	Peripheral Component Interface

PDPS	Planning and Data Processing System
PDR	Preliminary Design Review
PDS	Production Data Set
PDS	Production Data Specialist
PGE	Product Generation Executive
PGS	Product Generation System
PLNHW	Planning HWCI
POSIX	Portable Operating System for UNIX
PRONG	Processing CSCI
Q	Quarter
Q/A	Quality Assurance
QA	Quality Assurance
QAC	Quality and Accounting Capsule
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
REL	Release
RID	Review Item Discrepancy
RMA	Reliability, Maintainability, Availability
RTF	Rich Text Format
S/C	Spacecraft
SAA	Satellite Active Archives (NOAA)
SCF	Science Computing Facility
SCSI II	Small Computer System Interface
SDF	Software Development File
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment
SDPS/W	Science Data Processing Software
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SFDU	Standard Format Data Unit
SMC	System Management Center
SMP	Symmetric Multi-Processor
SPRHW	Science Processing HWCI
STMGT	Storage Management CSCI
TBD	To be determined

TBR	To be resolved
TDRSS	Tracking and Data Relay Satellite System
TONS	TDRSS Onboard Navigation System
TRMM	Tropical Rainfall Measuring Mission
TSDIS	TRMM Science Data and Information System
UR	Universal Reference
USNO	United States Naval Observatory
V0	Version 0
VC	Virtual Channel
VCDU-ID	Virtual Channel ID
WAIS	Wide Area Information Servers
WAN	Wide Area Network
WKBCH	Workbench CI
WKSHC	Working Storage HWCI
W/S	Workstation
WORM	Write Once Read Many
WS	Working Storage
WWW	World Wide Web